Cluster Computing (2020) 23:321–346
https://doi.org/10.1007/s10586-019-02924-2

Network adapter architectures in network on chip: comprehensive literature review

Babak Aghaei[1] · Midia Reshadi[2] · Mohammad Masdari[3] · Seyed Hadi Sajadi[4] · Mehdi Hosseinzadeh[5] · Aso Darwesh[6]

Received: 3 December 2017 / Revised: 2 January 2019 / Accepted: 9 March 2019 / Published online: 15 March 2019

Abstract
Network on Chip (NoC) is a new distributed, scalable, packet switched-based on chip which has been suggested as perfect solution for traditional centralized, non-scalable bus-based systems on chip (SoC) to handle issues like out-of order transactions, higher latencies, and end-to-end flow control. The NoC provides parallel and multi-core processing platform and is constructed from a set of Routers (R), Links (L), Intellectual Property (IP) cores, and Network Adapters (NA). The NA as individual hardware entity makes it possible IP cores with different data width and frequency connected to NoC. In other words, by decoupling computation from communication the NA allows IP Core modules and interconnects to be designed independently from each other. The design of NA impacts directly on NoC based SoCs critical parameters such as power dissipation, latency, throughput, and silicon area. This paper presents the comprehensive review of state-of-the-art architectures and the developments of NA which have been proposed in literature. Moreover, three type of parameters namely design (design goal, building components, Quality of Service (QoS), Core Interface Protocol (CIP), Security consideration, and Design for Test (DfT)), performance (power dissipation, latency, area, and throughput), and evaluation parameters (evaluation platform, clock frequency, technology scale) which have impact on NA architectures are evaluated and highlighted in comparative tables and figures. Furthermore, all the concepts that are considered in the design of NA is classified. Finally, concluding remarks and future research direction are provided.

Keywords Network on chip · Network adapters · Core interface · Network interface · Comprehensive review

1 Introduction

The number of cores which are able to be integrated into a single chip, as a System on Chip (SoC), has been increased because of the shrinking the size of transistor to sub-micron. In spite of this, because of leading bottleneck in communication system, this integration demonstrated that the hierarchical multi bus-based solutions are not scalable much more [1–5]. Therefore, the designers made decision to change the centralized, non-scalable bus-based systems on chip to new distributed, low power, scalable, reliable, services guaranteed, packet switched, and internet like layered protocol-based on chip networks that called Network on Chip (NoC) and as well as handling issues like out-of order transactions, higher latencies, and end-to-end flow control [6–17]. The NoC, which provides parallel and multi-core processing platform includes a set of Routers (R), Links (L), Intellectual Property (IP) cores, and

✉ Babak Aghaei
b.aghaei@iaut.ac.ir

1 Computer Engineering Department, Malekan Branch, Islamic Azad University, Malekan, Iran

2 Computer Engineering Department, Science and Research Branch, Islamic Azad University, Tehran, Iran

3 Computer Engineering Department, Urmia Branch, Islamic Azad University, Urmia, Iran

4 Telecommunication Research Center, Tehran, Iran

5 Iran University of Medical Sciences, Tehran, Iran

6 Information Technology Department, University of Human Development, Sulaymaniyah, Iraq

Network Adapters (NA) [18, 19]. The Routers are connected to each other with point-to-point Links due to one or more, homogeneous or heterogeneous Processing Elements (PE) more which is known IP cores, is clustered to a router via NA [20–22]. The NA is considered as an individual hardware entity that decouples computation and communication and makes it feasible to reuse IP core and communication infrastructure [23, 24]. NoC researches are classified into four areas: (1) system, (2) network adapter, (3) network and (4) link [16]. The flow of data through the network is indicated in Table 1 which demonstrates the relation between these research areas, the fundamental components of NoC, and the OSI layers [25].

The first level that is network aware is NA. The NA manipulates the end-to-end flow control, encapsulating the messages or transactions generated by the cores for the routing strategy of the network. These messages are broken into packets which containing information about their destination, or connection-oriented streams which do not, but have had a path setup prior to transmission. Two characteristics must be regarded to implement the NA [26]. The first one is utilizing standard interface protocol to fulfill the design reusability requirement and to reduce system design time. The second one is adopting an effective protocol conversion mechanism for better exploring the NoCs resources and performance to provide high-level communication services to the IP core by utilizing primitive services provided by the network hardware. Furthermore, the NA converts the Open Systems Interconnection (OSI) network layer services of the routers to transport and session layers services. Basically, Network Adapter design is important in the communication-centric system of Multi-Processor SoCs. In [27], a TV companion chip was redesigned with a NoC as interconnect, and 78% of increase in chip area was proved to come from the NAs. Similar case, in the Xpipes based system in [28], more than half of the NoC area is due to NAs. With the advent of NoCs, The designing NAs is faced with various challenges such as semantic gap between point-to-point protocols and network protocols, synchronization issues, packetization process, trade-off between packet length and initial transmission latency, flow control, and flit width selection [11]. There are several works for optimization of NA area, power consumption and delay which address one or more of the aspects such as low cost, high-level services offering an abstraction of the NoC, modular design, and differentiated services. But there is no work has been specialized in reviewing the NA architectures in NoC in detail. This paper presents the comprehensive review of state-of-the-art architectures and the developments of NA which have been proposed in literature. Moreover, design goals, evaluation methodologies, main components, and experimental performance measurements in NA architectures are discussed and highlighted. Furthermore, we discuss the challenges and classify all the concepts that are considered in the design of NAs. To the best of our knowledge this is the first scheme which tries to review NA architectures. The main contribution of this work are as follows:

- Presenting a comprehensive summarizing concepts in designing of Network Adapter.
- Providing a comparative analysis of the proposed approaches from aspect design, performance, and evaluation parameters.
- Classifying the previous proposed Network Adapter architectures into two main categories.

This paper organized as following: Sect. "Network Adapter Structure" presents Network Adapter structures. Section "Network Adapter Services" describes Network Adapter services. Comprehensive review of suggested Architectures for Network Adapter is located in Sect. "Classification of Proposed Network Adapter Architectures". In Sect. "Discussion", we discuss about the design parameters of previous works. Finally, the paper is ended with conclusion and future works. The acronyms used in this paper are defined as Table 2.

Table 1 Research area in network on chip

Research area	Fundamental components	OSI protocol level
System	Design methodology	Application
	Architecture domain	Presentation
	Traffic characterization	
Network Adapter	Functionality	Session
	Sockets	Transport
Network	Topology	Network
	Protocol	
	Flow control	
	QoS	
Link	Synchronization	Data Link
	Reliability	Physical
	Encoding	
	Data	

2 Network adapter structure

The NA implements a Core Interface (CI) at the core side (frontend) and a Network Interface (NI) at the network side (backend) [11]. The architecture of NA is depicted in Fig. 1.

The CI implements a standardized point-to-point protocol allowing IP cores reuse across several platforms. This

Table 2 List of acronyms

SoC	System on chip	DTL	Device transaction level
NoC	Network on Chip	GALS	Globally Asynchronous Locally synchronous
IP	Intellectual Property	ANOC	Asynchronous NoC
R	Router	FIFO	First In Firs Out
L	Link	QDI	Quasi Delay Insensitive
NA	Network Adapter	PU	Packetization Unit
CI	Core Interface	DU	Depacketization Unit
NI	Network Interface	DMA	Direct Memory Access
QoS	Quality of Service	TDM	Time Division Multiplexer
DfT	Design for Test	FF	Flip Flop
OSI	Open Systems Interconnection	LUT	Look Up Table
OCP	Open Core Protocol	MDNI	Direct Memory Network Interface
VCI	Virtual Component Interface	MPSoC	Multi-Processor SoC
AXI	Advanced Extensible Interface	MMR	Memory Mapped Register
GS	Guaranteed Service	FPGA	Field Programmable Gate Array
BS	Best Effort	ASIC	Application Specific Integrated Circuit
NSM	Network Security Manager	MANGO	Message Passing ANoC Guaranteed Service over OCP
DPU	Data Protection Unit	RDP	Resource Dependent Port
DoSP	Denial of Service Probe	RIP	Resource Independent Port
MNI	Master Network Interface	NISAR	NI Support Adaptive Routing
SNI	Slave Network Interface	CARS	Congestion Aware Request Scheduler

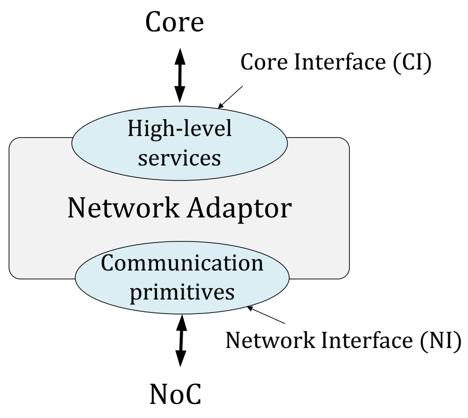


Fig. 1 Network Adapter architecture

solution allows core developers to focus on developing core functions, thus avoiding the need for advance knowledge regarding potential end-systems. The CI assumes the attributes of a socket, that is, an industry-wide Well-understood attachment interface which should capture all signaling between the IP core and the system (such as data-flow signaling, errors, interrupts, flags, software flow control, and testing). A distinctive requirement for this standard socket is to enable the configuration of specific interface instantiations along a number of dimensions (bus width, data handshaking, etc.). One commonly used socket is the Open Core Protocol (OCP) [29]. The OCP specification defines a flexible family of memory-mapped, core-centric protocols for use as a native core interface in on-chip systems. Another proposed standards such as Virtual Component Interface (VCI) [30] used in the SPIN [31] and Proteo [32] NoCs, Advanced eXtensible Interface (AXI) [33], Device Transaction Level (DTL) [34], and Wishbone [35] was also implemented in an NA design. In fact, the CI implements the services of OSI layering model's session layer. The session layer controls the transmission of data during the session, supports security, provides synchronization points in the stream of exchanged packets [24], and carries out the kind of transaction(s) that the IP core requests the targets.

The NI capsulates and encapsulates the packet, reorders buffers, implements synchronization protocols, and helps the router in terms of storage. Basically, the NI implements the transport layer services of OSI model. High-level communication services made available to IP cores at the session layer must be implemented by the transport layer, which is still unaware of the implementation details of the system interconnect. The transport layer relieves the upper layers from any concern with providing reliable, sequenced, and QoS-oriented data transfers. It is the first true and basic end-to-end layer [36]. For instance, the transport layer is in charge of establishing connection-oriented end-to-end communications [37, 38], as opposed to datagram-like communications [39], thus providing end-to-end control and information transfers with the QoS needed by the application program. The NI also provides, in its back-end, data-link layer services. Primarily, communication reliability has to be ensured by means of proper error-detecting strategies and effective error recovery techniques. Moreover, flow control is handled at this layer, by means of upstream (downstream) signaling regulating data arrival from the processor core (data propagation to the first switch in the route), but also of piggybacking mechanisms and buffer/credit flushing techniques.

Backward compatibility with mentioned protocols, is achieved by using a model based on transactions [40]. In a transaction-based model, there exist two types of IP cores: "Masters" and "Slaves". Masters initiate transactions by issuing requests, which can be further split in commands and write data. Examples of commands are "read" and "write". One or more Slaves receive and execute each transaction. Optionally, a transaction can also involve a response issued by the Slave to the Master to return data or an acknowledgement of transaction completion. This request-response transaction model directly matches the bus-oriented communication abstractions typically implemented at core interfaces, thus reducing the complexity of core wrapping logic [41]. Figure 2 shows the transaction model in NoC based SoC systems.

The basic NoCs such as Aetheral [42] offer their services on connections. The reason we use connections is to allow differentiated communication services and guarantees offered to the IP modules. Connections can be peer to peer (one Master, one Slave), multicast (one Master, multiple Slaves, all Slaves executing each transaction, and, currently, no responses allowed to avoid merging of messages), and narrowcast (one Master, multiple Slaves, a transaction is executed by only one Slave) [40], as shown in Fig. 3.

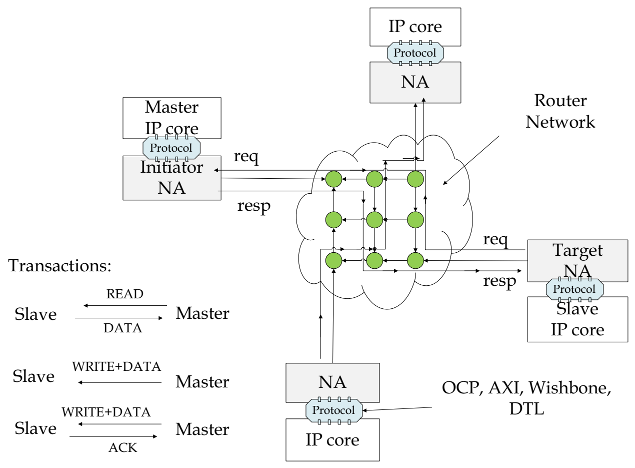


Fig. 2 Transaction model in Network on Chip

Connections are composed of unidirectional peer-to-peer channels (between a single Master and a single Slave). To each channel, properties are attached, such as guaranteed message delivery or not, in-order or unordered message delivery, and with or without timing guarantees. As a result, different properties can be attached to the request and response parts of a connection, or for different Slaves within the same connection. Connections can be opened and closed at any time. Opening and closing of connections takes time, and is intended to be performed at a granularity larger than individual transactions.

Message ordering is offered natively by the channel implementation. Within a channel, messages are packetized and sent by the NI in order of their receipt from the IP core. The packets in a channel are forced on the same path in the NoC, where they are kept in order by the routers and NIs. This choice limits the flexibility in routing the packets (e.g., no dynamic routing is possible), however, simplifies significantly NI design and reduces its cost, because there is no need to reorder within the NI. Ordering guarantees are provided only within a channel. Different channels are treated as separate entities in the NI scheduler, and they may have different routes. As a result, message reordering is possible across channels. Support for real-time communication is achieved by providing throughput and latency guarantees. These are essential for complex real-time streaming application, because they considerably reduce the integration time, and allow IP reuse. All connection properties that are end-to-end are implemented by the NAs. These are: reordering, transaction Guaranteeing, the Quality of Services in NoC completion, and flow control.

3 Network adapter services

The classification of the services offered by NAs in categories that have different design challenges is aimed in this section services span from session and transport-level services decoupling computation from communication, up to lower-level ones [43]. Here, these services is specialized in two main categories; CI Services and NI Services as Table 3.

The services that provided by NA is discussed in detail in the follow.

3.1 CI services

All services that the CI provided are called adaptation services. Their role is to adapt the communication protocol of the component to the communication protocol of the network. Of course, the challenge here is to minimize performance loss from a latency viewpoint.

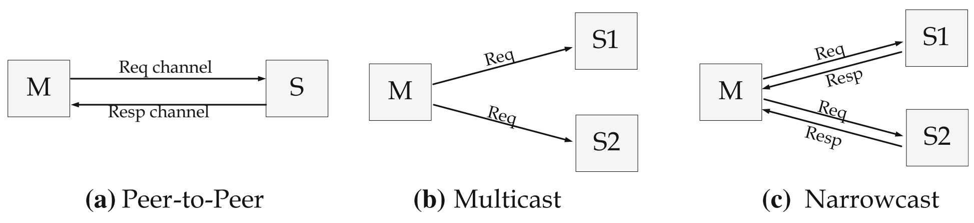


Fig. 3 Type of connections

Table 3 Network adapter services

Categories	Services	Sub services
CI Services	Adaptation Services	Core interfacing
		Packetization
		Clock adaptation
NI Services	Network Services	Bandwidth and latency guarantees
		Transactions ordering
		Flow control
		Reliable transactions
	Functional Services	Cache coherence
		Security and monitoring
		Test and Fault tolerant

3.1.1 Core interfacing

As we mentioned, the CI can be implemented to adhere to a SoC socket standard such as OCP, AXI, DTL, Wishbone and etc. With the right core interface design, the core can remain unchanged as it is reused in subsequent system designs that support that interface. By selecting an industry standard interface, there is no added time for this reuse approach since all cores require such an interface. This can dramatically accelerate final product delivery schedules.

3.1.2 Packetization

This is the very basic service the CI should offer: taking the incoming signals specifying processor core transactions, and building packets respective to the NoC communication protocol. This service should carefully optimize packet and flit size in order to achieve high-performance network operation and reduce implementation complexity of network building blocks. The amount of memory needed by the packetization service depends on the complexity of the packet size decision, on the packet structure, on the routing algorithm, and on the number of signals sampled at the interface with attached cores.

3.1.3 Clock adaptation

The main challenge of future SoCs id Clock distribution that is why future SoCs will probably be locally synchronous and globally asynchronous (GALS) [44]. Even if the clock frequency is the same over the chip, phase adaptation will be needed for communications [45, 46]. NoCs are composed of rather simple elements and rely on path segmentation, and thus they can potentially run at higher frequencies in order to decrease the latency seen by the networked computation units. In some NoC synthesis examples, it has been showed that the data-path can run at a higher speed than the control path [47] and that the critical path of a switch still lies in the arbitration logic and not in the interconnects [48]. Since the control path is slower than the data-path, a multi-phit flit allows the data-path to be kept busy while the switching control logic is taking a new decision. Alternatively, circuit switching also takes advantage of this decoupled design strategy [49].

3.2 NI services

The services that NI provides is divided in two service groups: network services and functional services.

3.2.1 Network services

The transport layer classical services are expected to implement that they implementation or not strongly depends on the features of the underlying network:

- Bandwidth and latency guarantees: If the NoC implements dynamic latency and/or bandwidth guarantees, then the wrapper has to manage this service. This means for example to build and send virtual circuit set-up or tear-down packets, or to allocate multiple buffeting resources and design complex packet schedulers in the

wrapper in order to handle traffic with different QoS requirements or maintain connections. Processor cores should not directly handle this, but they should be able to require, through the interface communication protocol with the NoC wrapper, transactions with different associated QoS levels, unless the association of processor core transactions with service classes has been statically performed.
- Transactions ordering: Reorder packets/transactions before forwarding them to the component (because sometimes in packet switched networks some packets arrive unordered which are sometimes unacceptable because of memory consistency issues). Transaction identifiers are used in such cases so as to know which packets should be completed in order.
- Reliable transactions: The NI could be involved in providing reliable network transactions by inserting parity check bits in packet tails at the packetization stage or by implementing end-to-end error control. Even in case link-level error control is implemented, the NI-to-switch link should be made robust with respect to communication errors. This might require special features of the flow control mechanism.
- Flow control: When a given buffeting resource in the network is full, there needs to be a mechanism to stall the packet propagation and to propagate the stalling condition upstream. The flow control mechanism is in-charge of regulating the flow of packets through the network, and of dealing with localized congestion. ACK/NACK, credit-based or on–off flow control are the most widely used strategies in NoC prototypes. The NI is involved with the generation of flow control signals exchanged with the attached switch. Moreover, packet flow control has to be carded out also with respect to the connected cores. In fact, when the network cannot accept new packets anymore because of congestion, new transactions can be accepted from the connected core just at the same, provided the necessary amount of decoupling buffeting resources is implemented in the NI. This mechanism allows to decouple (to a certain extent) core computation with its requests for non-blocking communication services. However, when buffers are full, the core behavior is impacted by the congestion in the network, since it has to be stalled if it requires further communication services.

3.2.2 Functional services

These services add new functionality to the system. In principle, many alternative implementations do exist for these functions (software, dedicated hardware, and processor core modifications), but in some cases their implementation in the NI achieves the best performance and allows design reuse.

- Cache coherence: This is a new issue brought by the introduction of NoCs. Indeed cache coherency can be achieved on a bus at very low cost by means of snoop devices, leveraging the shared nature of the communication medium. Cache coherence on a network is no longer an easy task because snooping is not possible. New protocols are therefore needed to allow the use of multiprocessor systems at low cost. This service may require cache modifications to allow access to their directories.
- Security and monitoring: Security-aware design of communication architectures is becoming a necessity in the context of the overall embedded SoC/device security. Complex communication infrastructure such as NoC may lead to new weaknesses in the system that can be critical and should be carefully studied and evaluated. On the other hand, NoCs can contribute to the overall security of the system, providing the ideal mean for monitoring system's behavior and detecting specific attacks [50, 51]. Probes, collecting information about the NoC traffic, are implemented inside Network Adapters [52–54]. In fact, NAs represent the ideal position where performing analysis of incoming traffic and discard malicious requests. A central unit is in charge of collecting security alerts and decide appropriate countermeasures.
- Low power: Early NoC prototypes are showing a significant contribution of the NoC to the system power dissipation. Beyond electrical- and gate-level low-power design techniques (e.g., deployment of low-swing signaling, low-power flip-flops, clock gating), higher-level techniques are likely to achieve larger savings. For instance, switching off some components and waking them up is one well-known system-level power management technique that could be applied also to network building blocks (especially NA).
- Test and fault tolerant: After its production process, NoC in its lifetime is exposed to a wide variety of faults which result in such malfunctions as data losing, efficiency degradation, and eventually, entire system breakdown. There is three policies in dealing with a fault; (1) detection, (2) location and (3) Fault tolerance. The faults, however, must be detected and located using test mechanisms before performing anything for NoC Fault tolerance [55]. Quite a number of investigations have been conducted on testing NoC components such as Routers [56–62], Communication channels [63–66], and IP Cores [67–70]. Furthermore, one of the most common test approaches is utilizing Built-In Self-Test (BIST). Generally, in more approaches, this mechanism

adds two hardware namely Test Pattern Generator (TPG) and Test Response Analyzer (TRA) to Network Adapter. Moreover, in fault tolerant approaches [71], the mechanisms such as fault detection and diagnosis codes, retransmission, Triple Module Redundancy (TMR), need to be realized in NA designs as individual hardwares too.

4 Classification of proposed network adapter architectures

The proposed architectures for the Network Adapter blocks is classified in Fig. 4. As indicated in this figure, the proposed architectures are categorized into Buffer-based (including FIFO-based and DMA-based) and Transaction-based (including OCP-compliant, AHP-compliant, DTL-compliant, Wishbone-compliant, VCI-compliant, and AXI-compliant).

4.1 Buffer-based NA architectures

Buffer-based architectures use mostly a memory structure as buffer in their design. In this approaches, IP core only uses the handshake and credit signals to communicate with NoC. Buffer-based architectures include FIFO-based and DMA-based architectures which are described in continue.

4.1.1 FIFO-based NA architectures

In [4], Liang et al. try to connect IP cores to each other in a single chip interconnect architecture that is called adaptive System on Chip (aSoC) by a simple Network Adapter. In this approach, IP cores are effectively associated with extra hardwares (control gates and FIFOs) that is named “Interface” to form a computational node. The

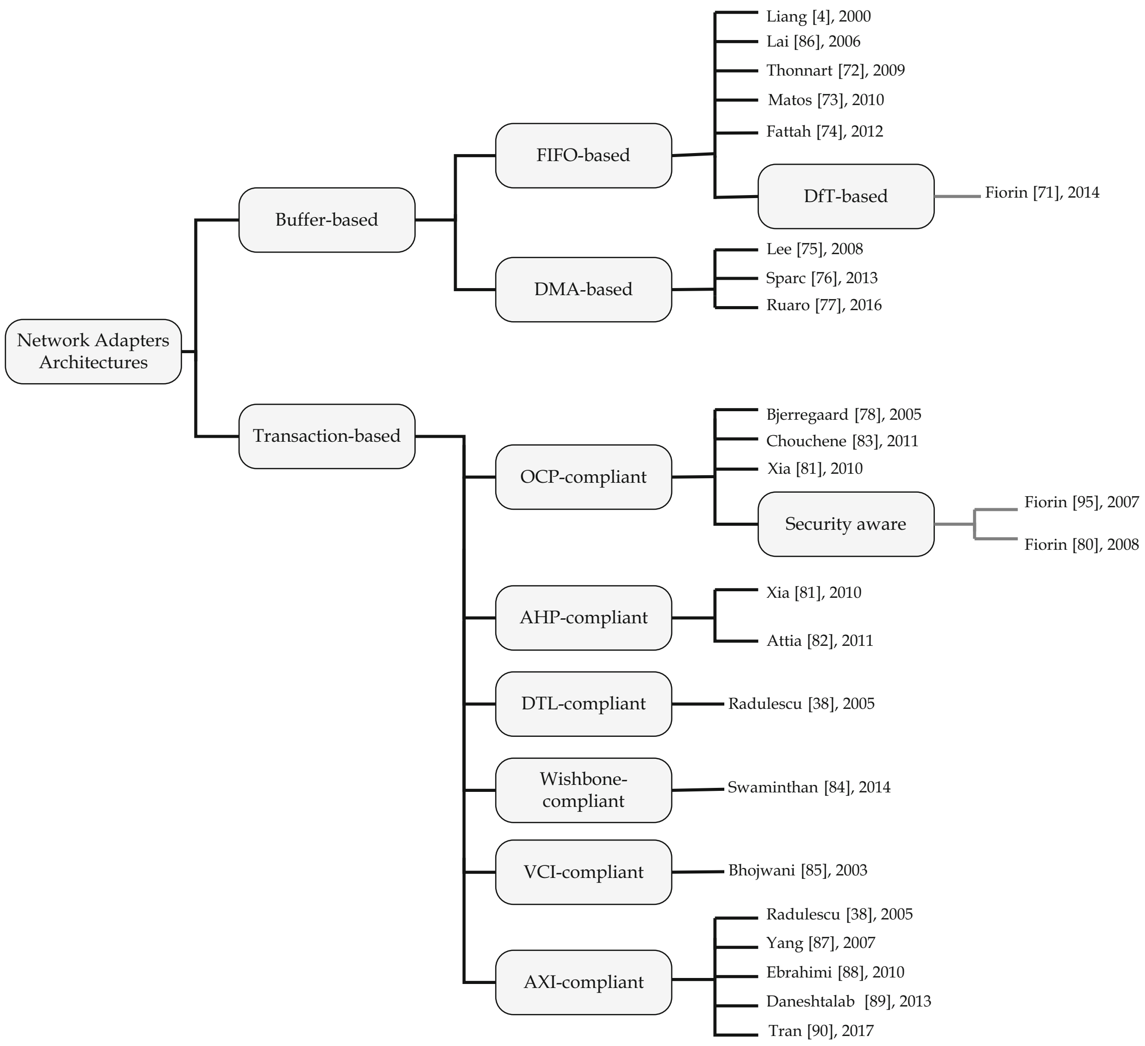


Fig. 4 The classification of proposed NA architectures

communication between nodes takes place via pipelined, point-to-point connections. Figure 5 depicts the proposed architecture.

This approach supports for compile-time, scheduled communication and allows for scalable bandwidth across a range of heterogeneous IP cores. The evaluation results show interfacing approach is up to 5 times more efficient than bus-based SoC interconnect architectures and can be developed compactly so that it can be integrated into a variety of possible cores. Additionally, it can accommodate IP cores of varying sizes, aligned in a mesh like structure. The significant limitation of the current approach is the lack of support for packetization/depacketization, arbitration and routing.

Another FIFO-based design of NA Gglobally Asynchronous Locally Synchronous adapter (GALS) to be used in Asynchronous Network on Chip (ANoC) architecture proposed by Thonnart et al. in [72]. The proposed NA is a complete IP integration module, including two new FIFO (S-A and A-S), and a local programmable clock generator and a delay line units. The design of proposed NA is depicted in Fig. 6.

The FIFOs based design using a Johnson encoding principle applied for timing domains interfacing. The programmable delay line element used in the local clock generator. The latches offer a good compromise in terms of delay and energy, while filtering out all unnecessary pulses within the delay line. Only one path through the delay line is activated according to the programmed binary value. The conversion between synchronous and Quasi Delay Insensitive (QDI) asynchronous logic was completely rethought in order to minimize the latency added by 4-phase conversion, and leads to an important gain in performance respectively to the previous solutions. The complete GALS adapter provides programmable clock frequencies up to 1 GHz, for an asynchronous throughput of more than 500 MHz. The experimental results illustrate the two FIFOs with a width of 36 bit and a length of 5 occupy 2500 μm^2 each, while the A-S and S-A ANoC adaptation layers use respectively 1100 μm^2 and 2300 μm^2. The local

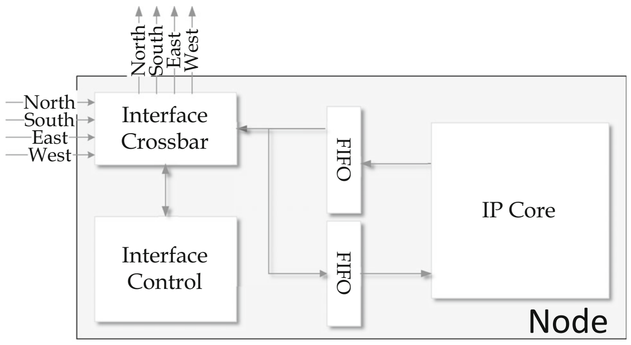


Fig. 5 The proposed approached in [4]

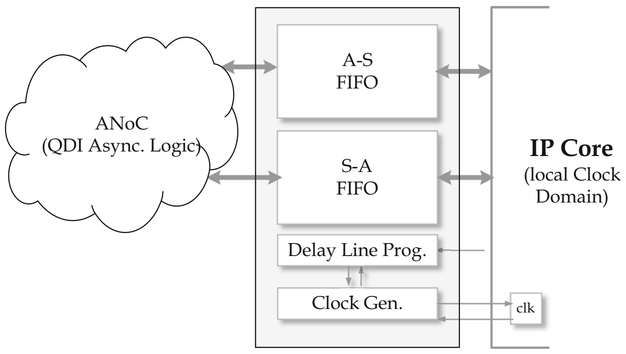


Fig. 6 The proposed NA in [72]

clock generator is relatively small in comparison with about 500 μm^2.

In [73], Matos et al. present a solution to the synchronization problem in NoCs. They demonstrate an NA with the synchronizer is needed for asynchronous and also synchronous communication architectures. The proposed NA consists two main blocks: “Synchronizer Wrapper” and “Pack and Unpack Interfaces”. Furthermore, it is designed to associate packets of different sources that need to be placed in sequence. Thus, they claim solved the problem of synchronization present in data flow applications. The NAs is designed with flexible network configuration and, besides, a single set of unpack and pack NAs can be shared for multiple cores. The synthesis results illustrate the area of NA standard by itself is 18,735 μm^2, the NA standard + Synchronizer is 50,019 μm^2, and NA standard + Synchronizer + Memories is 169,817 μm^2.

A new NA, Transport-NI (Tra-NI) with knowledge about the transport layer protocols is presented by Fattah et al. in [74]. Tra-NI was developed for dynamic NoC-based many-core systems using message passing paradigm.

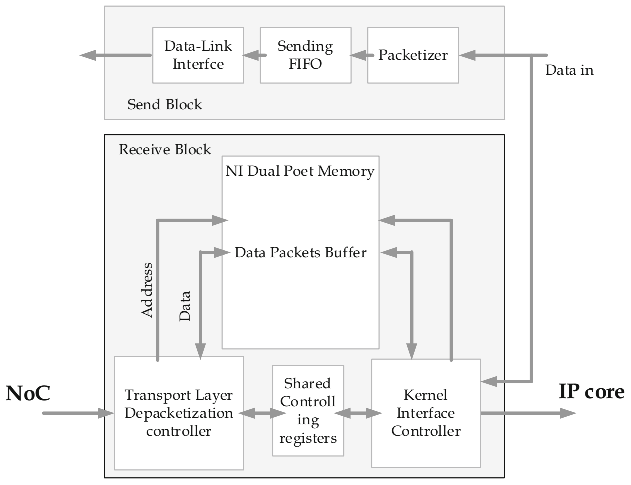


Fig. 7 The proposed NA in [74]

The Tra-NI has two block namely send block and receive block as shown in Fig. 7.

The Tra-NI's receive block contains a dual port memory block for storing arrived packets, instead of the kernel. The memory is divided into two parts. A large portion is used as buffer for data packets (packets which exchange application data) and a smaller part, for control packets (commands and reports to the kernels), is manipulated as FIFO. The depacketizing controller goes through the transport packets (encapsulated in the network packets), analyzes the header, and takes the appropriate action. It decodes the transport header to indicate the packet type. Data packets are stored in a free place in the buffer, while other packets are injected into the FIFO part of the memory. The controller searches for a free place for the next data packet after the current data packet has been completely delivered. As a result, the next packet might not need to wait for finding an empty place. If the buffer is full after receiving the current packet, the search mechanism is started as soon as a packet is consumed by the kernel. The depacketizing controller stops receiving packets by means of wormhole switching when the memory is full until a free place is available. Whenever a receive function is executed, the kernel requests Tra-NI to lookup the buffer for the packet with the desired properties. If the interface controller finds the requested packet it returns the packet to the kernel. Otherwise, it informs the kernel about the failure, and stores the request in its internal registers. Respectively, the kernel either copies the data part of the packet directly to the task memory, or suspends the task until its requested packet arrives in the Tra-NI. When a new packet arrives, the interface controller checks among the pending requests. If the packet is already requested, the kernel is informed about the packet arrival through the interrupt signal. Then the kernel copies the data to the destination task memory and resumes the task execution. Of note, the kernel is informed upon the packet arrival which diminishes the penalty of store and forward policies; i.e. wor mhole switching. Furthermore, the interface controller interrupts the kernel upon the arrival. However, the proposed NA eases the networking job of kernels and reduces their performance bottleneck. This is done in the receiver side of the NA by depacketizing, storing, and retrieving transport packets in the hardware level. The extracted results showed some benefits over original NAs that are implemented in the network layer.

4.1.2 DMA based NA architectures

In [75], Lee et al. provide a generic architecture for NA and associated wrappers for a NoC based multiprocessor SoC in order to allow systematic design flow for accelerating the design cycle. For a generic NA, they classify the possible IP cores for PE and introduced an allocation table for a wrapper design. The allocation table is used for the configuration of the modular wrapper and for the software adaptation. Basically, the NA consists of a “packetization unit (PU”), a “depacketization unit (DU) a”nd “PE interface”. Figure 8 depicts the architecture of proposed NA.

The PU consists of a header builder, a flit controller, a send Direct Memory Access (DMA) controller and registers. Moreover, the PU builds the packet header and converts the data in the memory into flits. The header builder forms the packet header based on the information provided by registers such as destination address, data ID, number of body flits and service level. DMA controller generates control over the address and read signal for the internal memory by referring the start address of the memory and the number of data for block data/program transfer. Flit controller wraps up the head flit and body flits into a packet. The DU includes a flit controller, a header parser, a DMA controller and registers. It performs the receiving data from interconnection network. The flit controller selects head flit from a packet and passes it to the header parser. The header parser extracts control information from the head flit such as address of source PE, number of body flits, and specific control parameters. Also, it asserts an interrupt signal to the IP core to get the local memory address for the packet. DMA controller automatically writes the body flit data into the internal memory. According to experimental results, the proposed NA including two FIFOs has an area of approximately 0.053 mm^2 (NI Area + FIFO Area $\times$ 2) using the 90 nm technology.

In [76], Spars et al. presented a novel and area efficient network interface for a Time Division Multiplexer (TDM) based NOC. The key ideas are to use the already dual-ported communication memories (SPMs) for clock domain crossing, and to move the Dynamic Memory Access (DMA) controllers from the processor cores into the NAs. This avoids the need for buffering and flow control, it allows the DMA controllers to directly deliver the payload

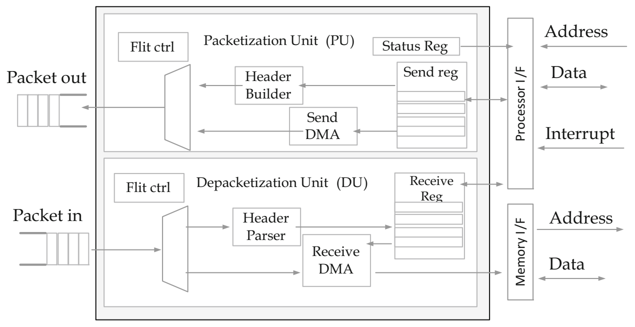


Fig. 8 The proposed NA in [75]

data to outgoing packets in accordance with the TDM schedule, and it opens for a very interesting and efficient table based implementation of the DMA controllers. The block diagram of proposed architecture, shown in Fig. 9.

The key elements of this architecture are the slot counter, the slot table, and the DMA table. The slot counter is reset and incremented in all NAs using the same clock and the slot counter defines the slots in the TDM schedule period. A slot corresponds to the time it takes to transmit a packet into the NOC, and consequently the number of clock cycles in a slot is identical to the number of flits in a packet. The slot counter indexes a slot table whose entries consist of a valid bit and an index into the DMA table. The valid bit indicates whether or not a packet is to be sent in the corresponding time slot. The interfaces denoted “M” and “S” are master and slave ports supporting point-to-point read/write transactions. An entry in the DMA table holds all the registers that are found in a normal DMA controller: some control bits (start and done), a read pointer, a write pointer, and a word count. In addition to this, an entry also holds the route to be used when a packet is sent across the NOC. The reason to split between a slot table and a DMA table is to be able to assign more than one slot in a TDM schedule period to a connection and reserve different amounts of bandwidth for different end-to-end circuits. A variation of the architecture involves storing the route information in the slot table. This would allow a connection that has been allocated several slots in the TDM period to use different routes in the different slots. Another variation merges the slot table and the DMA table into a single table. This may allow a more efficient hardware implementation, and it is possible if no connection is assigned more than one slot within the TDM period. The presented NA is synthesized with a slot period of 32 and 8 DMAs per NA to a 90 nm CMOS process. Using standard cells and flip-flop-based tables the total area of a NA is 0.024 mm^2, and the breakdown is NA logic 0.0063 mm^2, slot Table 0.0036 mm^2 and DMA Table 0.014 mm^2.

In [77], Ruaro et al. presented a specialized communication NA for NoC-based Multi-Processor SoCs (MPSoCs), called DMNI (Direct Memory Network Interface). The DMNI merges the functionalities of the DMA and the NI into a single component, directly connecting the NoC router with the processor memory. Figure 10 details the DMNI architecture.

The DMNI contains three main modules: “send”, “receive”, and “arbiter”. The arbiter manages the memory accesses for both modules, enabling simultaneous send and receive operations. The µkernel controls the DMNI through memory mapped registers (MMRs). The DMNI design is generic because it enables to send and receive any type of data. To avoid stalls in the communication, the design of the DMNI supports simultaneous packet reception and transmission. The adoption of the DMNI reduced the latency to transmit packets, execution time, and area (Field Programmable Gate Array (FPGA) and Application Specific Integrated Circuit (ASIC) technology) when compared to the baseline implementation (DMA + NI). A simplified and generic programming interface exposes the DMNI services to the software layer. This integration removed unnecessary interfaces, registers, and signals, being specialized for NoC-based MPSoCs. Results show a reduction in the silicon area and performance improvement in the packet transmission. Based on synthesis results, comparing the baseline design (DMA + NI) with the proposed DMNI, there is a small area reduction (3.47%) when targeting an ASIC implementation. On the other side, for FPGAs, an important reduction in the number of flip-flops is observed—48%, with an increased number of LUTs—11.5%. The reduction observed in the number of flip-flops comes from the smaller number of registers required by the DMNI implementation.

4.2 Transaction-based NA architectures

Transaction-based architectures are complained with one of the standard IP core interface protocol such as OCP,

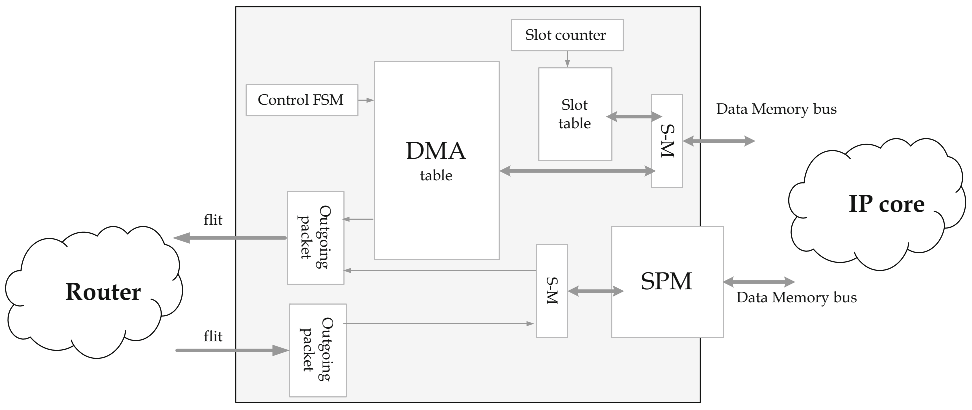


Fig. 9 The proposed NA architecture in [76]

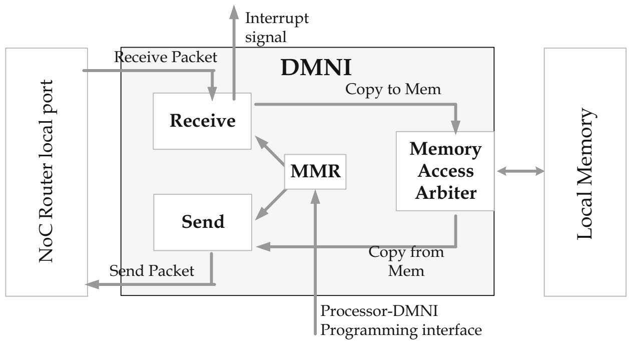


Fig. 10 The proposed NA architecture in [77]

VCI, AHP, Wishbone, DTL, and AXI. Transaction-based architectures include OCP-compliant, AHP-compliant, DTL-compliant, Wishbone-compliant, VCI-compliant, and AXI-compliant NA architectures.

4.2.1 OCP-compliant NA architectures

In [78], Bjerregaard et al. propose an OCP compliant NA architecture for the Message-passing Asynchronous NoC providing Guaranteed Services (GS) over OCP interfaces (MANGO) [79]. In MANGO, IP cores reside in locally clocked regions, connected by a heterogeneous network through NAs. The NAs use the primitive message-passing functionality implemented by the network, to transparently provide the IP cores with read/write transactions. MANGO provides connection-less best-effort (BE) routing, as well as connection-oriented guaranteed services, in terms of hard latency and bandwidth bounds. Key features of MANGO are (i) clockless implementation, (ii) guaranteed communication services and (iii) standard socket access points.

A clockless implementation of the network routers and links enables a GALS type system, in which the integration of cores with different timing characteristics is inherently supported. An advantage of clockless circuits is that they use zero dynamic power when idle. Also, their forward latency can be made much lower than that of comparable clocked circuits, helping to minimize the performance overhead of using a network for e.g. memory accesses. Read/Write transactions require two types of NAs: an initiator connecting to a master core such as a microprocessor, and a target connecting to a slave core such as a memory. Figures 11 and 12 illustrate the structural design of the initiator and target NAs.

It is also seen how each NA consists of a request and a response path, and–orthogonally to this–of a clocked and a clockless part. The presented version of the MANGO NA supports all the required OCP v2.0 basic signals and a consistent subset of burst, thread and interrupt extensions, allowing support for single reads and writes, single-request

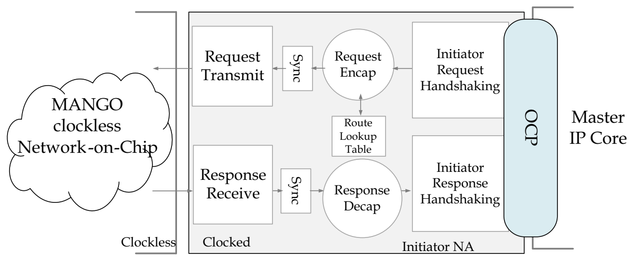


Fig. 11 Initiator NA proposed in [78]

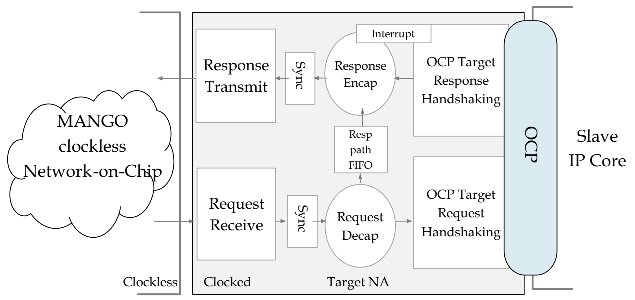


Fig. 12 Target NA proposed in [78]

burst reads and writes, threads, connections and interrupts. The NAs provide a number of input and output network. This approach shows due to the GS input buffers, the area of the target NA supporting GS ports is twice that supporting only a BE port. The NAs with only a BE port are simpler than those with multiple GS ports, hence their port speed is higher.

In [80] Fiorin et al. present a monitoring system for NoC based architectures, whose goal is to help detect security violations carried out against the system. Information collected are sent to a central unit for efficiently counteracting actions performed by attackers. The monitoring system is mainly composed of three elements: Probes (P), a Network Security Manager (NSM), and the Communication Infrastructure (NAs and Routers). Probes are located within NAs. The choice of embedding them inside NAs presents several advantages such as analyzing the traffic, performing parallel monitoring by kernel for OCP interface, limiting the malicious traffic. Figure 13 depicts proposed NA architecture.

Two probes rely on the presence of a Data Protection Unit1 (DPU) embedded inside the NA. First, the Illegal Access Probe (IAP) detects attempts to illegally access restricted memory blocks or range of addresses in shared memory systems, while the Denial of Service Probe (DoSP) is employed to detect Bandwidth Reduction or Draining Attacks. The Event Generator is triggered by the two probes and generates the packets to be sent to the NSM to communicate the security violations. The DPU is a hardware block enabling the access to a given memory space only if the initiator of the request is authorized to do

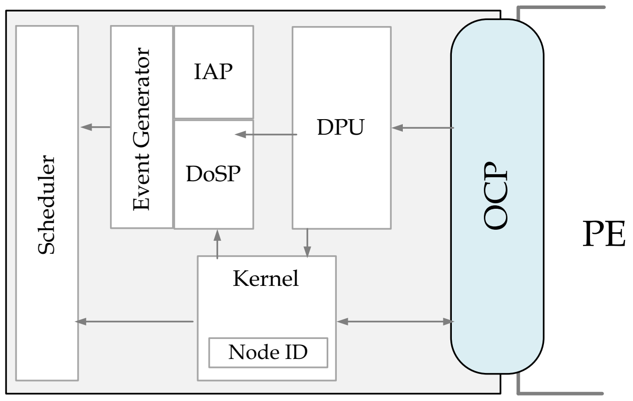


Fig. 13 The proposed NA in [80]

the operation. Access filtering is performed by considering not only the memory address, but also type of operation requested (data load/store), and the status (role) of the initiator (user or supervisor mode). Embedded in the initiator's NA, the DPU acts as a firewall in data networks, stopping requests not allowed in the targeted memory blocks or peripherals. A dedicated core, the NSM, is in charge of collecting events and information coming from the several probes distributed in the system, analyse the data received, and counteract efficiently to the detected attacks. While focusing on hardware characteristics of the monitoring system, it leaves to software designers the task of implementing detection strategies for security violations and appropriate countermeasures. Traffic produced by probes is to be maintained separated from standard communication traffic inside the NoC. In order not to be sensitive to Daniel of Service (DoS) attacks addressing NoC performance, the transmission of packets from probes must be guaranteed through the use of priority communication or guaranteed throughput services. Moreover, differently from the case of on-chip debug, messages coming from probes should not be accessible by external entities through public interfaces, in order to avoid the exploitation of the information collected to attacks the system. The synthesis results demonstrate the overall security system, including the DPU and the two probes, counts for around 25.6% of the NA implementation that in compared to a NA implementation without security monitoring, the overhead associated is around 34.7%.

An another new NA design which refer to as mutual interface definition based method suggested by Xia et al. in [81]. It decouples Resource Dependent Part (RDP) from Resource Independent Part (RIP) by mutual interface definition. The RDP consists of two units: "configure unit" and "data adapter unit". The configure unit is used to configure the control interface signals. The data adapter unit is designed to adapt data signals compliant with IP cores to those compliant with data interface. The RIP adopts a configurable structure to provide deployment flexibility based on the signals of control interface. The RIP takes care of multiple functions, including network communication protocol implementation, end-to-end flow control, data packetization, depacketization, multi-clock domains synchronization, etc. The RIP is composed of five main components: a protocol controller unit, a packetization unit, a depacketization unit, an input buffer and output buffer. These two parts can be designed independently, thus the design flexibility and reusability of NA can be enhanced. Moreover, a NA component library consisting of multiple RDP and RIP components is proposed to be built. The networks-on-chip designers can choose appropriate components from the library to construct NA design. From the perspective of RDP, the NA achieves backward compatibility with the existing protocols such as AMBA AHB and OCP. From the perspective of RIP, the NA provides a configurable structure supporting multicast transfer and adaptive routing algorithm extensions.

A study on the implementation of fault-tolerant NAs for NoCs is carried out by Fiorin et al. in [71]. By performing a fault injection campaign on the NoC, NAs, and routers, they demonstrated how the NA could be the main source of errors in the NoC, in particular when the number of nodes in the network increases. Figure 14 shows the basic functional blocks of the NA taken as reference in their evaluations.

The occurrence of permanent and temporary faults in the NA could cause an unwanted behavior that may create unrecoverable situations in the NoC, such as deadlock or livelock conditions. So, the Authors propose a functional fault model for the NA based on the behavior on its main components, i.e., the lookup table, FIFOs, and the finite state machines driving NA operations and proposed new architectural solutions based on the use of error correcting and detecting codes and a limited amount of redundancy, and discussed policies for the reconfiguration of the components that should be applied at the detection of errors. The experimental results demonstrate a saving of up to 48% in the area overhead is optained, as well as a

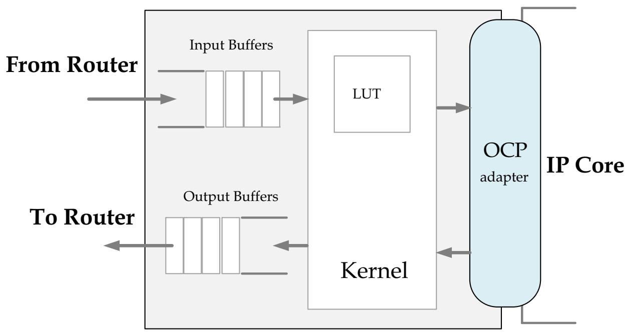


Fig. 14 The proposed NA architecture in [71]

significant energy reduction, with respect to an alternative standard hardware Triple Module Redundancy (TMR) implementation of the NA, while maintaining a similar level of robustness to faults.

4.2.2 AHP-compliant NA architectures

In [82], Attia et al. present a pipelined NA architecture that implement the services provided by the protocol layer of OSI model at a low cost. The distributed buffer structure implemented in proposed Network interfaces allow to reduce the jitter between two successive packets and allows NA to work without blocking fashion. The authors design two types of NA for AHB based cores for our network-on-chip, named Master NA attached to master IP and Slave NA attached to slaves IP. Each type of NA is additionally split in two sub modules, one for the request and one for the response data flow or channel (injection and extraction path). Figures 15 and 16 depict proposed internal architecture of two NAs.

In Master NA, there are two fundamental separations in the NA architecture that enable this modularity: a horizontal one which distinguishes the injection path (request data flow) from the extraction path (response data flow), and a vertical one which distinguishes between the network-dependent and the network independent (connected component) part. Separation between injection and extraction functions allows easy reuse of dual components in both master and slave NAs, since injection corresponds to packet composition and transmission, while ejection corresponds to packet reception and decoding. The tasks of the SNI are to receive request packages from the network, decapsulate the request packages, transmit the request to the slave core, receive response from the slave core, encapsulate response and transmit response to the network. Clearly, the proposed architecture of the slave network interface is built on two path. The extraction path performs the transformation of the request packet of our NoC to an AHB request. The injection path performs the transformation of the AHB response to a response packet to NoC. The physical division of the interface is distributed in two parts: Shell and Kernel. The Shell part communicates with slave IP respecting the AHB protocol.

The author of [82] in another work [83] try to reduce the power dissipation of NoC by reducing the NA power. They present a hardware design of a low power NA design for both Master NA and Salve NA. The low power is obtained by the implementation of a mechanism based on stoppable clock technique for power saving. The stoppable clock technique allows us to shut down each sub module when it is not under running. The synthesis study shows that the proposed architecture reduces 63% and 37% in terms of dissipated power relatively to the initial Master NA respectively for handshake and credit based control flow with area overhead of 19% and 2%. The average decrease of the maximum frequency is about 21%.

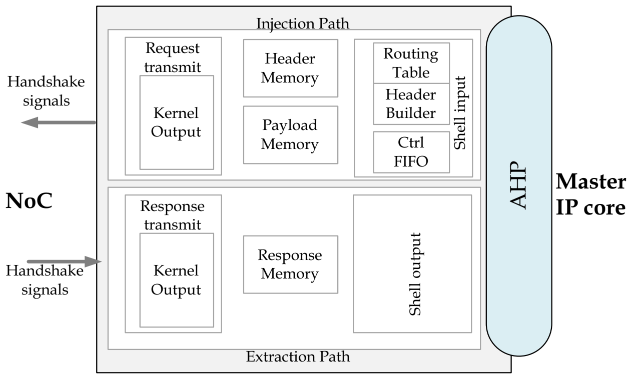


Fig. 15 Master NA proposed in [82]

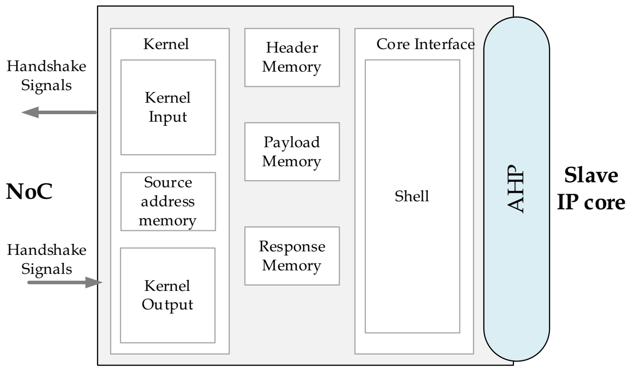


Fig. 16 Slave NA proposed in [82]

4.2.3 DTL-compliant NA architectures

A new NA architecture is developed by Radulescu et al. in [38] which offers high-level services(guaranteed and best-effort services) at a low cost and provides a shared memory abstraction, where communication is performed using Read/Write transactions. Authors offer, via connections, high-level services, such as transaction ordering, throughput and latency guarantees, and end-to-end flow control. These connections are configurable at runtime via a memory-mapped configuration port. They use the network to configure itself as opposed to using a separate control interconnect for network configuration. Proposed NA has a modular design, composed of a kernel and several shells as Fig. 17 is illustrated.

The NI kernel provides the basic functionality, including arbitration between channels, transaction ordering, end-to-end flow control, packetization, and a link protocol with the router. Shells implement: (1) additional functionality, such as multicast and narrowcast connections and (2) adapters to

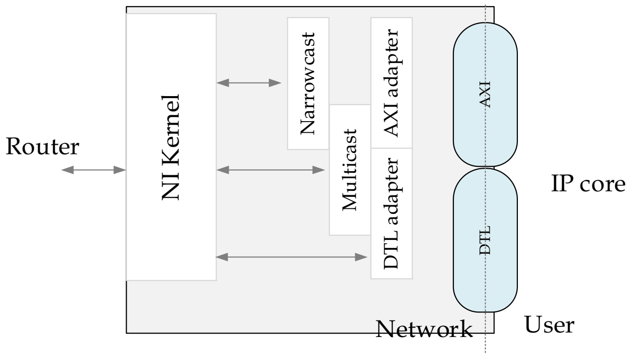


Fig. 17 The proposed NA architecture in [38]

existing protocols, such as AXI or DTL. All these shells can be plugged in or left out at design time according to the needs. They illustrate an instance of proposed NA, which shows that the cost of implementing proposed protocol stack in hardware is small. This approach illustrates the cost of implementing this approach in hardware is 0.25 mm^2 after layout in a 0.13 micron technology, running at 500 MHz.

4.2.4 Wishbone-compliant NA architectures

In [84], Swaminathan et al. with the aim to speed up the data transfer in the NA present a generic asynchronous FIFO-based Wishbone compatible plug and play NA for NoC design. The authors believe the existing AMBA, OCP and DMA-based NAs utilizing many types of handshake and credit based flow control offer high latency, low throughput and low speed. So, they present a new NA architecture that offers lower latency due to latency free wrappers with merged micro-level architecture of one clock cycle latency packing/unpacking modules and optimum latency of one clock cycle asynchronous FIFO compared to the existing designs. Figure 18 depicts the block diagram of suggested NA.

According to Fig. 18, the Master/Slave interface module at the router side generates the required signals to write the data into the FIFO and read out the data from the packing module which generates the required signals to translate the packing module signals to Wishbone compatible signals and also generates the necessary control signals to write the data from NoC router to FIFO. Similarly the master/slave interface module at the processing core side generates the required signals to write the data into the asynchronous FIFO and read out the unpacked data from the unpacking module and generates the required signals to translate the unpacking module signals to Wishbone compatible and generates the necessary control signals to write the data from processing core to asynchronous FIFO. This interface will perform single/block read/write operations on both sides. The Wishbone reset signal is active LOW and is used to reset the IPs and the bus. Experimental results show that the proposed NA offers a low latency of 2 clock cycles, 4.5 times higher throughput when compared to the best available ASIC implementation and 33.76% more compared to the best available FPGA implementation. The speed of the proposed NA is increased by 170% in ASIC design and 16% increase in FPGA based design and the area is reduced by 10% in ASIC, 52% and 12% reduced in number of slice registers and LUTs in FPGA based design.

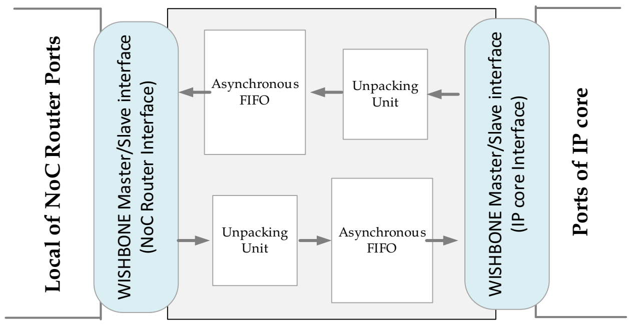

Local of NoC Router Ports
WISHBONE Master/Slave interface (NoC Router Interface)
Asynchronous FIFO
Unpacking Unit
Unpacking Unit
Asynchronous FIFO
WISHBONE Master/Slave interface (IP core Interface)
Ports of IP core

Fig. 18 The proposed NA architecture in [84]

4.2.5 VCI-compliant NA architectures

In [85], Bhojwani et al. also try to connect IP cores to on-chip interconnection network through developing a wrapper that has three components, namely; the packet assembly, packet transmission and the packet disassembly and delivery. In this approach, on-chip network consists of the basic building block, the tile. The tiles or clients are connected to a network that routes packets between them. Each tile may consist of one or more cores (processor cores, memory cores, etc.). The tile would have routing logic, which would be responsible for routing, forwarding the packets, based on the routing policy of the network. For cores that are neither programmable nor reconfigurable, the only option for interfacing with the networking logic of the tile is to utilize a wrapper, which would have the responsibility of packetizing and depacketizing the cores requests and responses. The wrappers have the responsibility of (i) receiving the contents from the core interface, preparing the packets and dispatching them to the network logic of the tile and (ii) receiving the packets from the networking logic and presenting the contents to the core interface. The authors examine the latency characteristics in the packet assembly stage of the on-chip communication. The different packetization strategies that have been investigated in this paper are Software library based, On-core module based, Wrapper based. According to simulation result, Software implementation results in high latencies and hardware wrapper implementation has the lowest area overhead and the lowest latency.

In [86] Lai et al. propose a NA design suitable for popularization employ which supports serial-link packet-based transmission model for NoC application. The NA function consists of packet construction, flow control, error detect; re-transmission mechanism, data scrambler and 8b/10b transformation. The whole function is partitioned into three layers and designed as a peer-to-peer architecture. Figure 19 depicts the layer-based NA architecture.

Every layer can be further divided into a transmitting portion that processes outbound traffic and a receiving portion that processes inbound traffic. This proposed NA aims to reduce the link loading, provide transmit data in serial and avoid the data transmission skew.The layout design of the NA demonstrate the occupied area on silicon is about 0.43 mm in a 0.13 micron CMOS technology. The working frequency is about 312.5 MHz.

4.2.6 AXI-compliant NA architectures

An AXI compliant NA for NoC suggested by Yang et al. in [87] which can deals with the reordering problem and support the adaptive routing. On the basic of analyzing the necessity and feasibility of packet reordering, they propose a reordering mechanism based on look up table, which can guarantee globally ordering of the response transactions. Their proposed NI architecture Supporting Adaptive Routing (NISAR) supports the master and slave core together. Figure 20 depicts the component of proposed architecture for NA.

As shown in the Fig. 20, the proposed architecture has three parts as follows:

(1) The packetization datapath, which transmit the data from the AXI interface to the router. It performs the arbitration between write and read transaction channels, master port (MP) and slave ports (SP); implements credit based flow control, and packetize the transactions into packets. The Master IP and Slave IP deliver the request or response transaction to NA, through MP and SP. Before a transaction is packetized in Packetizer unit, it should be granted by MP (SP) and by Pack Scheduler unit. After the packetization, packets will be sent to the VC Pre-election unit to choose an output Virtual Channel of router based on the credits.

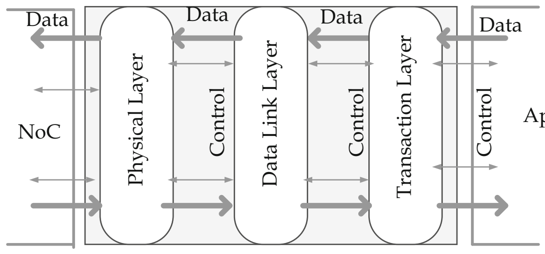


Fig. 19 The proposed NA architecture in [86]

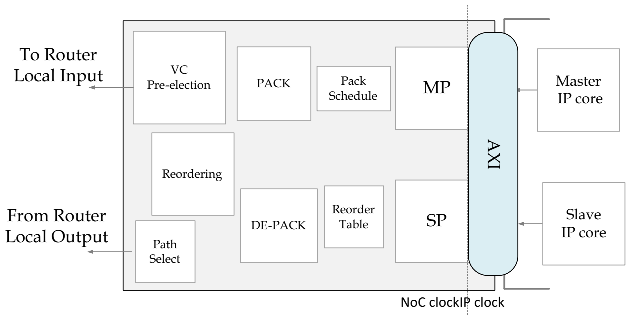


Fig. 20 The proposed NA in [87]

(2) The de-packetization datapath, which receives the packets from the router and converts them back to AXI transactions, has 3-stage pipeline. When every packet header enters the NA, Path Select unit decides which unit the whole packet should enter. If the request packet header comes, it will be sent the De-pack unit directly. If the response packet header comes, the seq_id and Tran_id will be used to look up the reorder lookup table. If it is the expected packet, it could de-packetize directly, and otherwise it will be sent to the Reordering unit to wait. When payloads entered NA, the decision will not change until another header coming from same direction. De-pack can depackatize four different packets at the same time, then deliver the 4 group AXI transactions to the MP or the SP through a crossbar.

(3) Reorder part includes two units (Reordering and Reorder Table) to support the reordering function of NISAR. It records the order of the packets, which is the most important for the reordering. It provide the Pack unit the seq_id within the packet header; it helps Path Select unit to decide which packet should wait for reordering, which should be de-packetized directly; it also tell the Reordering unit when these packets can be de-packed.

The synthesize result shows that NISAR runs in 150 MHz. The area for NISAR is 0.28 mm^2, in which Reordering unit takes 0.15 mm^2, corresponding to 53.6% of NISAR area.

In [88] the authors present a new NA architecture for on-chip networks to increase the resource utilization and to improve overall performance. The proposed approach aims to solve deadlock via the reordering mechanism that is exploited by the NA. The resource utilization of the conventional reordering methods is not efficient enough. Also, based on the proposed architecture, a hybrid NA is presented to integrate both memory and processor in a tile. The hybrid model is formed by combining the master-side and slave-side network interfaces. The architecture of hybrid NA is shown in Fig. 21. The proposed architecture exploits AXI transaction based protocol to be compatible with existing IP cores.

Based on the type of incoming packet the detector unit determines the target unit (Slave-side Queue/Master-side Queue). Regarding the MPSoC's configuration, if each node is supposed to integrate a dedicated processor and memory, instead of using two NA (master and slave), the hybrid model is more beneficial, particularly in terms of area and power costs. According to synthesis result, a hybrid NA model reduces 14.3% and 13.7% in hardware area and power dissipation respectively.

In [89], Daneshtalab et al. present Congestion-Aware Request Scheduler (CARS) approach to deal with the network congestion. The main idea of CARSs is to utilize global congestion information to reduce sending packets to the congested area by prioritizing packets in slave NAs. Packets collect the congestion information along paths and carry it to slave NAs. However, the information of nearby routers are sufficient to estimate the global congestion of different regions. The reason is that, the information of faraway routers is outdated when it arrives to the source node. Based on this perspective, in CARS, packets start collecting information when their distances are equal or less than three hops from the destination node. Each slave NA maintains a table to store the estimated congestion value carried by packets. To evaluate the performance of the proposed schemes, uniform and non-uniform/localized synthetic traffic patterns are considered. These workloads provide insight into the strengths and weaknesses of the CARS method in the congestion-aware on-chip networks. In the non-uniform mode, 70% of the traffic is local requests, where the destination memory is one hop away from the master core, and the rest 30% of the traffic is uniformly distributed to the non-local memory modules. Furthermore, the adaptive scheduler scheme in the slave NA imposes 9% hardware overhead while offering 23% performance gain.

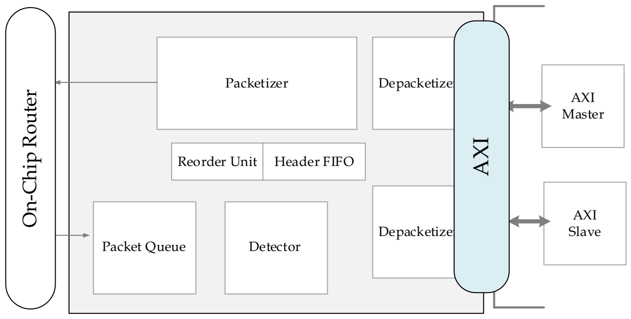


Fig. 21 The proposed NA in [88]

In [90], Tran et al. have presented the design and implementation of a AXI compliant NA, which is used to adapt ARM cores into NA the NoC architectures. It supports the source routing algorithms, Wormhole switching communication mode in 2D/3D NoC with different topologies. In this approach, two types of NA to implement the synchronization between AXI interfaces and the NoC switching fabric, named Master NA (MNA) and Slave NA (SNA). The MNA is used to create the connection between an AXI compliant master IP core and a network router while a slave IP core connects to a network router through the SNA. Each NA consists of two sub-modules: Request Flow unit and Response Flow unit. In the AXI protocol, the most important operation is the burst transaction. It enables the use of bandwidth more effectively. The proposed NAs is supporting three AXI burst-based transaction types: fixed, incremental, and wrapped burst. On one hand, the Master Network Adapter (MNA) encapsulates requests from the master IP core into network packets which are sent to the network. On the other hand, it also decapsulates responses from the routers and transfer them to the master IP cores. The MNA mainly composes of two data flows: MNA Request Flow and MNA Response Flow, which are responsible for encapsulation and decapsulation One of the advantages of the AXI-NoC adapter is that it offers a simple NA architecture, which requires low area footprint to implement the design, while achieves a high communication throughput and low power consumption. Another advantage of the design is the usage of the mux-selection method to decide whether the outputs to one-side interface are controlled by the state machine or directly by the input handshaking signals from the other side. Based on synthesis results, at 650 MHz, the area cost of the design is 952 gates and the total cell area is 2.793 mm^2. The design can achieve a communication throughput of 20.8 Gbits/s (i.e., 650 Mflits/s $\times$ 32 bits/flit), and the NA consumes approximately 4.14 mW.

5 Discussion

The current research paper aims to comprehensively classify the NA architectures as well as presenting a comparative analyze of that from the critical parameters in NoC based SoCs aspects. To do so, the authors faced with the following challenges:

- Different names such as Network Interface (NI), Core Interface (CI), IP core-NI interface, and Network Adapter (NA) were utilized to refer to this unit. The name Network Adapter (NA) was used to provide a unity in the naming. It propose the future works use the Network Adapter term for naming that it has to main block; Network Interface (NI) and Core Interface (CI).

- Some research works are assumed the NA is as a part of Tile [91, 92] or as a part of IP core [93]. So, analyzing and reviewing these works are difficult. At the end of 2010, the NA is assumed as an individual and stand-alone component between IP core and Router.
- The most works do not obtain the full simulation result.
- The most authors do not explain the reasons for choosing the CI protocol.
- Our intention to bring the number of articles is to indicate the extent to which the volume of articles is directed toward each other. Our look at the number never ignored us in evaluating the amount of each parameter. We have evaluated the parameters in the various tables.
- None of the work on the hardware implementation platform has been mentioned.
- Most previous studies have not referred to implementation specifications such as network dimensions, network type, network topology, packet injection rates, package sizes, and so on.

Despite these challenges, we classify all NA designs to two main categories; Buffer-based and Transaction-based. Figure 22 depicts the percentage of these two categories relative to total designs.

The Fig. 2 demonstrate the number of Transaction-based NA architectures is more than Buffer-based architectures, according to the following reasons, the researchers accept the Transaction-based approaches:

- Transaction-based approaches use standard interface to establish the strong connection and guaranteeing protocol layer services.
- The NA can be either proprietary or standard. A proprietary interface reduces the reusability of the network but may improve its performance. A standard interface has opposite characteristics [94].

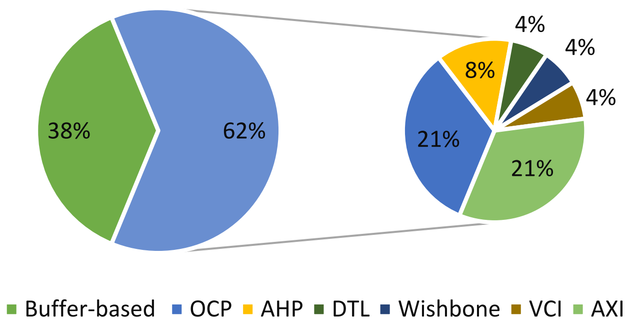


Fig. 22 The percentage of Buffer-based and Transaction-based architectures

- By implementation of standard interface in NA design the router adaptation is not necessary. This simplifies the design process and can reduce the NA area overhead.
- This approaches allow IP core developers to focus on developing core functions, thus avoiding the need for advance knowledge regarding potential end-systems.

Totally, the comparison of proposed architectures is investigated based on three aspect of parameters namely; "design", "performance", and "evaluation" parameters. In continue, we present the result of comparisons.

5.1 Design parameters

The considered design parameters are as following:

- The goal of design
- Building components
- Quality of service
- Core Interface Protocol(CIP)
- Security consideration
- Design for test

The design parameters of proposed NA approaches is evaluated in the Table 4. Briefly, the goal of design is the main target authors to present the paper (column 2). Building components includes the main blocks of NA that is explained in every paper (column 3). Column 4 determines the applied CI protocols. Whether the works is discussed about QoS is brought in columns 5. The security and data protection of works are surveyed in columns 6. There is only two works that focuses on security. In the last column, the paper determines which works are investigated the Design for Test polices that there is only a work.

The reason that some of the cells are empty is that the authors do not present sufficient information about these parameters. For better analysis, the Fig. 23 is extracted from Table 3 that shows the number of proposed NA approaches which are considered the CIP, QoS, security, and DfT parameters.

It is evidential that although QoS, Security and DfT are important parameters for obtaining high performance in sub-micron technologies [96–99], these are overlooked in the reviewed NA architectures.

5.2 Performance parameters

In this paper, we are considered performance parameters such as:

- Area
- Power dissipation
- Latency
- Throughput

Table 4 Comparative analysis of the design parameters in proposed NA architectures

Schemes	Design goal	Building components	CI Protocol (CIP)	QoS	Security	DfT
[4]	Scalable Communication Interface	FIFO, crossbar, interface control				
[85]	Interface the IP Core to communication network	Packet assembly, packet transmission and the packet disassembly and delivery	VCI	✔		
[38]	NA with GS and Shared-Memory Abstraction	Kernel, shell	AXI DTL	✔		
[78]	OCP compliant NA	Core Interface(CI), Network Interface(NI)	OCP	✔		
[86]	High speed NA	PLP, MUX, Scrambler, 8b/10b encoder, 8 to1 PISO				
[95]	Network Interface (with Data Protection Unit)	Data Protection Unit	OCP	✔	✔	
[87]	AXI compliant NA	Packetization datapath, de-packetization datapath, Reorder part	AXI			
[75]	Network Interface (generic architecture)	Packetization unit (PU), a depacketization unit (DU) and PE interface				
[80]	Network Interface with Security Monitoring Service	Probes(P), Network Security Manager (NSM), communication infrastructure	OCP		✔	
[72]	NA for ANoC	A-S and S-A buffers, clock generation.		✔		
[88]	Hybrid NA in a tile	AXI-Queue, Packetizer/Depacketizer, Reorder, detector, header FIFO	AXI			
[81]	NA(mutual Interface)	Resource dependent part (RDP), resource independent part (RIP)	AHB OCP			
[73]	Synchronizer NA	Synchronizer, packing, depacking				
[82]	Pipelined NA (latency and jitter optimization)	Request channel, response channel	AHB			
[83]	Low power Network Interface (Credit Based)	Control FIFO, Header builder, Network Interface, Stoppable clock	OCP			
[74]	Transport Layer Aware NA Design	Receive Block, send Block				
[89]	Congestion-Aware Request Scheduler	AXI-Queue, Packetizer, Packet-Queue, Depacketizer, Reorder	AXI			
[76]	Area-efficient NA for TDM-based NoC	Scratchpad memories, slot counter, slot table, DMA table		✔		
[71]	Fault-tolerant Network Interfaces design	Lookup Table (LUT), FIFOs, Finite State Machines (FSMs).				✔
[84]	Wishbone-based Network interface	RX/TX Async FIFO, Packing/unpacking units	Wishbone			
[77]	Unified design (DMNI)	Send Module, Receive Module, Memory Access Arbiter				
[90]	AXI based Network Adapter(with High-Performance Adaptation)	Request Flow Unit and Response Flow unit	AXI			

The performance properties of reviewed NA architectures are analyzed in Table 5. In column 2, the area of NA architectures is specified with the occupied silicon area that are obtained by synthesis tools. In some cases, the occupied area is determined by count of LUTs and FF. The power dissipation refer to overall amount of power consumption of NA including components in Watt or Joule (column 3). The time that NA delayed to prepare the response named latency. In column 4, the latency of proposed NAs specified by Clock, Second and Cycle. The amount of networks throughput that are applied the NA architecture is illustrated in last column.

Figure 24 depicts the count of previous works that are considered the performance parameters; power dissipation, latency, area and throughput. It is worth to note, the most proposed NA approaches are investigated the area parameter and a few of them focus on the throughput parameter.

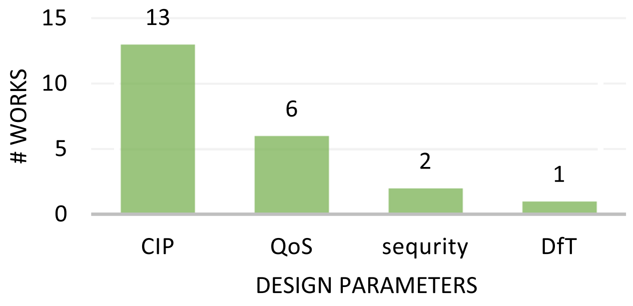


Fig. 23 The comparison of design parameters in proposed architectures

5.3 Evaluation parameters

The evaluation parameters are defined as following items:

- Evaluation platforms
- Clock frequency
- Technology scale

Table 6 summarizes the evaluation parameters of proposed NA approaches with values. Evaluation platforms includes all simulation and implementation tools which are applied by authors (column 2). In column 3, the semiconductor technology scales is used by evaluation platforms are specified. The clock frequency is maximum territory that allows to the evaluation platform to get correct results (last column). Form Table 5, it is undrestood that the technology scale varies between 45 nm to 1.2 µm and the clock ferquency varies between 100 MHz to 2 GHz.

The number (left side) and the percentage of evaluation platforms (right side) which have been exploited by the proposed NA architectures are specified in Fig. 25. Around 28% of reviewed works i.e. 9 works are used Synopsis Design toolset [100] for evaluation of proposed NA designs.

Similarly, in Fig. 26, in left side, the number and in the right side, the percentage of Nano-scale technologies which have been used by proposed NA architectures are illustrated.

It is notable, the authors mostly are used 130 nm and 45 nm is the smallest scale in CMOS technology that is

Table 5 Comparative analysis of the performance parameters in proposed NA architectures

Schemes	Area		Power/Energy (Watt/Joule)	Latency (Clock/second/cycle)	Throughput
	LUTs	FFs			
[4]	0.50 mm^2			8.5 ns, 3 clock	
[85]				3.02 ns	
	4 K				
[38]	0.25 mm^2			4–10 cycles	
[78]	0.097 mm^2			6 clock	
[86]	0.43 mm^2			10 clock	
[95]	0.54 mm^2		506.6 pJ		
[87]	0.28 mm^2			3–4 cycle	0.87 flits/cycle
[75]	0.053 mm^2		27.5 mW		
[80]	2433 µm^2		407.6 pJ		
[72]	6.4 µm^2		12.7 mW	2 clock write	714 Mflits/s
[88]	101 mm^2		6.12 µW		
[81]	0.13 mm^2		×	4 cycles	
[73]	170 mm^2		8.24 mW		
[82]	0.054 mm^2		14.5 mW	3 cycles	
[83]			74 mW	4 cycles	
	393	602			
[74]				287 clocks	1563 Mbs
	623				
[89]	0.0491 mm^2		24 mW		
[76]	0.024 mm^2				
[71]	0.723 mm^2		0.371 pJ		
[84]	32,526 µm^2		32 mW	2 cycles	32 Gbs
[77]	21.4 mm^2			116	
	761	409			
[90]	2.8 mm^2		4.14 mW	1 cycle	20.8 Gbs

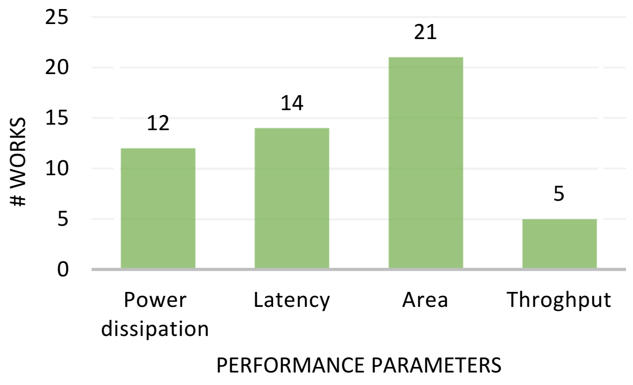


Fig. 24 The comparison of performance parameters in proposed architectures

applied to implementation of the proposed NA architectures.

6 Conclusion

The Network on Chip provides parallel and multi-core processing platform and composed of a set of Routers (R), Links (L), Intellectual Property (IP) cores, and Network Adapters (NA). The NA is an individual hardware entity that decouples computation and communication and make it feasible to reuse IP core and communication infrastructure. The NA implements a Core Interface (CI) at the core side (frontend) and a Network Interface (NI) at the network side (backend). The CI implements a standardized point-to-point protocol allowing IP cores reuse across several platforms. The NI capsulates and encapsulates the packet, reorders buffers, implements synchronization protocols, and helps the router in terms of storage. The NA provides two main group services; CI services and NI services. All services that the CI provided are called adaptation services. Their role is to adapt the communication protocol of the component to the communication protocol of the network. Of course, the challenge here is to minimize performance loss from a latency viewpoint. The services that NI provides is divided in two service groups: network services and functional services. In network services, the transport layer classical services are expected to implement that they implementation or not strongly depends on the features of the underlying network: Bandwidth and latency guarantees, Transactions ordering, Reliable transactions, Flow control. In functionality services, these services add new functionality to the system. In principle, many alternative implementations do exist for these functions (software, dedicated hardware, and processor core modifications), but in some cases their implementation in the NI achieves the best performance and allows design reuse.

Table 6 Comparative analysis of the evaluation parameters in proposed NA architectures

Schemes	Evaluation platforms	Technology scale	Clock frequency
[4]	NSIM simulator, SPICE	1.2 µm	333 MHz
[85]	Synopsys Design Analyzer	0.35 µm	200 MHz
[38]	Philips internal design flow	0.13 µm	500 MHz
[78]	OCP tool Core Creator	0.13 µm	400 MHz
[86]	CMOS technology	0.13 µm	312.5 MHz
[95]	STMicroelectronics technology library	0.13 µm	500 MHz
[87]	TSMC technology	0.13 µm	150 MHz
[75]	Synopsys, TSMC	90 nm	719 MHz
[80]	STMicroelectronics technology library	0.13 µm	500 MHz
[72]	ST CMOS technology, Synopsys	65 nm	500 MHz
[88]	2D NoC simulator, Synopsys D.C.	90 nm	2 GHz
[81]	TSMC CMOS standard, Synopsys	90 nm	1.35 GHz
[73]	ModelSim, Synopsys Design	0.180 µm	126 MHz
[82]	Synopsys Design	0.13 µm	833 MHz
[83]	FPGA, XPower		462 MHz
[74]	HeMPS 3.9, Xilinx		100 MHz
[89]	Synopsys Design, Cadence SoC Encounter	90 nm	1 GHz
[76]	CMOS library, Altera Cyclone chip	90 nm	
[71]	Synopsys Design Compiler	45 nm	500 MHz
[84]	Xilinx	90 nm	2 GHz
[77]	FPGA (Xilinx XC5VLX330)/ASIC(65 nm)	65 nm	
[90]	CMOS technology	45 nm	650 MHz

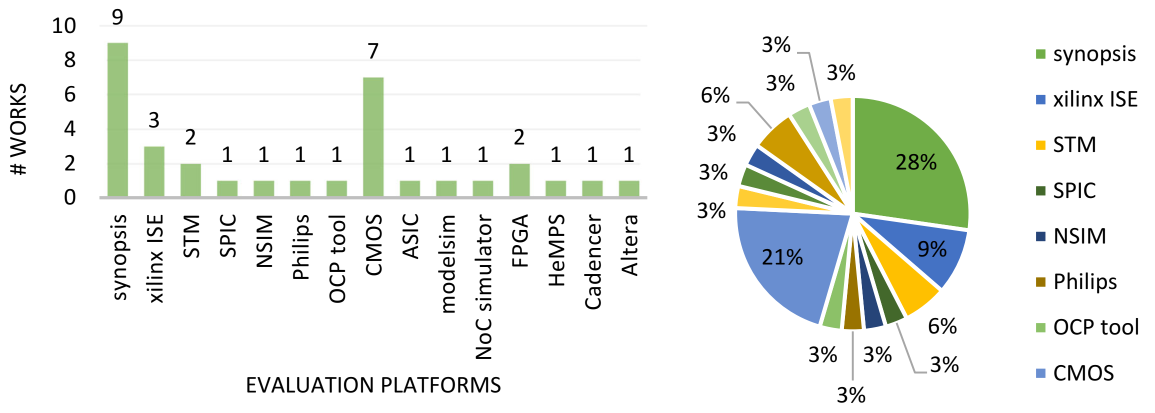
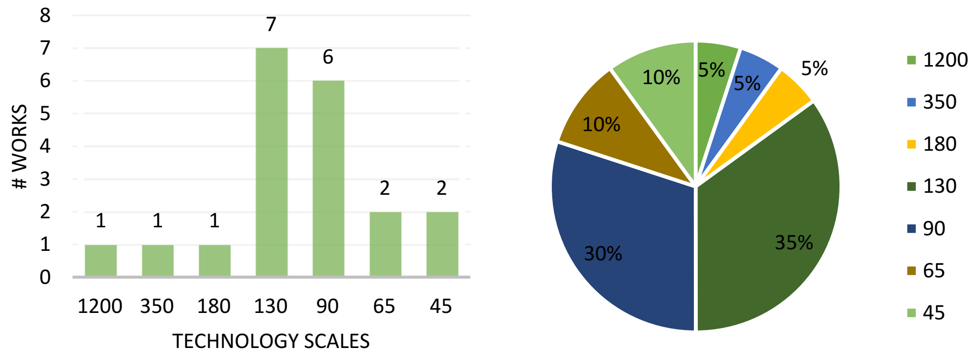


Fig. 25 The evaluation platforms applied by authors

Fig. 26 The technology scales



Basically, Network Adapter design is important in the communication-centric system of Multi-processor SoCs. Moreover, The designing Network Adapters is faced with various challenges such as semantic gap between point-to-point protocols and network protocols, synchronization issues, packetization process, trade-off between packet length and initial transmission latency, flow control, and flit width selection. To the best of our knowledge this is the first work which tries to review NA architectures. The proposed architectures are classified into buffer-based (including FIFO-based and DMA-based) and transaction-based (including OCP-compliant, AHP-compliant, DTL-compliant, Wishbone-compliant, VCI-compliant, and AXI compliant). Into these two categories, the paper identifies the architectures which realizes Design for Test (DfT) and security considerations. The NA architectures that use a memory structure as buffer in its design is called Buffer-based architectures. The IP core only uses the handshake and credit signals to communicate data with NoC. The transaction-based proposed architectures are applied one of the standard protocol such as OCP, VCI, AHP, Wishbone, DTL, and AXI for guaranteeing the data transmission in IP core and NA connection. Based on comparison result, the transaction-based approaches are mostly accepted by researchers for the reasons given below:

- Transaction-based approaches use standard interface to establish the strong connection and guaranteeing protocol layer services.
- The NA can be either proprietary or standard. A proprietary interface reduces the reusability of the network but may improve its performance. A standard interface has opposite characteristic.
- By implementation of standard interface in NA design the router adaptation is not necessary. This simplifies the design process and can reduce the NA area overhead.

Moreover, three type of parameters namely design (design goal, building components, quality of service, Core Interface Protocol (CIP), Security consideration, and design for test), performance (power dissipation, latency, area, and throughput), and evaluation parameters (evaluation platform, clock frequency, technology scale) which have impact on Network Adapter architectures discussed and highlighted. In this paper, the result of classification and comparison provided analytical figures and tables.

According to the outcoms of the present study, the following suggenstions are provided for future researches:

1. Future work will involve the analysis of traffic behaviors of possible attacks and the integration of

the monitoring system with software strategies to detect possible security threats.
2. Try to adapt IP cores with more standard interfacing protocols.
3. By adding hardware units, the NoC designers try to make component of NoC more testable and fault tolerant. As the future works, one can focuses on the test and fault tolerant approaches.
4. Future articles are intended to be more effective, and the parameters to be considered in design, performance, and evaluation (13 in total parameters) are also included.
5. There is a few work that are considered quality of service in NA. So as future work, we expect to see the NA architectures that are handle the guarantee of service (GS) and best effort (BE) atributies.

To see the behaviour of proposed NAs in high performance and giga-scale application it is required the authors run the evaluation platfroms with high frequencies to obtain real results.

6. It is expected the future works to implement its proposed mechanism on a scalable NoC and report the evaluation parameters (power consumption, delay, throughput).

References

1. Chang, K.-C., Shen, J.-S., Chen, T.-F.: Evaluation and design trade-offs between circuit-switched and packet-switched NOCs for application-specific SOCs. In: Proceedings of the 43rd annual Design Automation Conference, ACM, pp. 143–148 (2006)
2. Pasricha, S., Dutt, N.: On-chip communication architectures: system on chip interconnect. Morgan Kaufmann, Burlington (2010)
3. Zeferino, C.A., Kreutz, M.E., Carro, L., Susin, A.A.: A study on communication issues for systems-on-chip. In: Integrated Circuits and Systems Design, 2002. In: Proceedings of the 15th Symposium, IEEE, pp. 121–126 (2002)
4. Liang, J., Swaminathan, S., Tessier, R.: aSOC: a scalable, single-chip communications architecture. In: Parallel Architectures and Compilation Techniques, 2000. In: Proceedings of the International Conference, IEEE, pp. 37–46 (2000)
5. Dehyadgari, M., Nickray, M., Afzali-Kusha, A., Navabi, Z.: A new protocol stack model for network on chip. In: Proceeding of the IEEE Computer Society Annual Symposium Emerging VLSI Technologies and Architectures, IEEE, p. 3 (2006)
6. Agarwal, A., Iskander, C., Shankar, R.: Survey of network on chip (noc) architectures & contributions. J. Eng. Comput. Archit. **3**(1), 21–27 (2009)
7. De Micheli, G., Benini, L.: Networks on chip: a new paradigm for systems on chip design. In: Proceeding of the IEEE Computer Society on Design, Automation & Test in Europe Conference & Exhibition, pp. 0418–0418 (2002)
8. Tatas, K., Siozios, K., Soudris, D., Jantsch, A.: Designing 2D and 3D network-on-chip architectures. Springer, New York (2014)
9. Benini, L., De Micheli, G.: Networks on chips: a new SoC paradigm. Computer **35**(1), 70–78 (2002)
10. Grecu, C., Ivanov, A., Saleh, R., De Micheli, G.: Design, synthesis, and test of networks on chips. (2005)
11. De Micheli, G., Benini, L.: Networks on chips: technology and tools. Academic Press, Cambridge (2006)
12. Poluri, P., Louri, A.: A Soft Error Tolerant Network-on-Chip Router Pipeline for Multi-core Systems
13. Kumar, S., Jantsch, A., Soininen, J.-P., Forsell, M., Millberg, M., Oberg, J., Tiensyrja, K., Hemani, A.: A network on chip architecture and design methodology. In: VLSI, 2002. In: Proceedings of the IEEE Computer Society Annual Symposium, IEEE, pp. 105–112 (2002)
14. Jantsch, A., Tenhunen, H.: Networks on chip, vol. 396. Springer, New York (2003)
15. Goossens, K., Dielissen, J., van Meerbergen, J., Poplavko, P., Rădulescu, A., Rijpkema, E., Waterlander, E., Wielage, P.: Guaranteeing the quality of services in networks on chip. In: Networks on chip. pp. 61-82. Springer, (2003)
16. Bjerregaard, T., Mahadevan, S.: A survey of research and practices of network-on-chip. ACM Comput. Surv. (CSUR) **38**(1), 1 (2006)
17. Hemani, A., Jantsch, A., Kumar, S., Postula, A., Oberg, J., Millberg, M., Lindqvist, D.: Network on chip: An architecture for billion transistor era. In: Proceeding of the IEEE NorChip Conference (2000)
18. Dally, W.J., Towles, B.P.: Principles and practices of interconnection networks. Elsevier, Amsterdam (2004)
19. Henkel, J., Wolf, W., Chakradhar, S.: On-chip networks: A scalable, communication-centric embedded system design paradigm. In: Proceedings 17th International Conference on VLSI Design, IEEE, pp. 845–851 (2004)
20. Ramanujam, R.S., Soteriou, V., Lin, B., Peh, L.-S.: Design of a high-throughput distributed shared-buffer NoC router. In: Proceeding of the 2010 Fourth ACM/IEEE International Symposium on Networks-on-Chip (NOCS), IEEE, pp. 69–78 (2010)
21. Chan, C.-H., Tsai, K.-L., Lai, F., Tsai, S.-H.: A priority based output arbiter for NoC router. In: Proceedings of the Circuits and Systems (ISCAS) on IEEE International Symposium, IEEE, pp. 1928–1931 (2011)
22. Saponara, S., Vitullo, F., Petri, E., Fanucci, L., Coppola, M., Locatelli, R.: Coverage-driven verification of HDL IP cores. In: Conti, M. (ed.) Solutions on embedded systems, pp. 105–119. Springer, New York (2011)
23. Sgroi, M., Sheets, M., Mihal, A., Keutzer, K., Malik, S., Rabaey, J., Sangiovanni-Vencentelli, A.: Addressing the system-on-a-chip interconnect woes through communication-based design. In: Proceedings of the 38th annual Design Automation Conference, ACM, pp. 667–672 (2001)
24. Bertozzi, D.: Network interface architecture and design issues. Networks on Chips: Technology and Tools, The Morgan Kaufmann Series in Systems on Silicon, pp.147–202 (2006)
25. Zimmermann, H.: OSI reference model–The ISO model of architecture for open systems interconnection. IEEE Trans. Commun. **28**(4), 425–432 (1980)
26. Wang, J., Yang, Z.J.: Design of network adapter compatible OCP for high-throughput NOC. In: Applied Mechanics and Materials, pp. 1341–1346. Trans Tech Publ (2013)
27. Steenhof, F., Duque, H., Nilsson, B., Goossens, K., Llopis, R.P.: Networks on chips for high-end consumer-electronics TV system architectures. In: Proceedings of the conference on Design, automation and test in Europe: Designers' forum, European Design and Automation Association, pp. 148–153 (2006)

28. Angiolini, F., Meloni, P., Carta, S., Benini, L., Raffo, L.: Contrasting a NoC and a traditional interconnect fabric with layout awareness. In: Proceedings of the conference on Design, automation and test in Europe, European Design and Automation Association, pp. 124–129 (2006)
29. Alliance, O.: Open core protocol specification. In. Release, (2003)
30. Alliance, V.: Virtual component interface standard. http://www.vsi.org/library/specs/summary.html. (2001)
31. Guerrier, P., Greiner, A.: A generic architecture for on-chip packet-switched interconnections. In: Proceedings of the conference on Design, automation and test in Europe, ACM, pp. 250–256 (2000)
32. Ahonen, T., Sigüenza-Tortosa, D.A., Bin, H., Nurmi, J.: Topology optimization for application-specific networks-on-chip. In: Proceedings of the 2004 international workshop on System level interconnect prediction, ACM, pp. 53–60 (2004)
33. ARM, A.: AXI Protocol Specification, version 1.0 www.arm.com, ARM. In. March, (2004)
34. Radulescu, A., Dielissen, J., Goossens, K., Rijpkema, E., Wielage, P.: An efficient on-chip network interface offering guaranteed services, shared-memory abstraction, and flexible network configuration. In: Proceedings of the Design, Automation and Test in Europe Conference and Exhibition, IEEE, pp. 878–883 (2004)
35. Opencores, S.: Wishbone system-on-chip (soc) interconnection architecture for portable ip cores. http://cdn.opencores.org/downloads/wbspec_b3.pdf (2002)
36. Tanenbaum, A.S.: Computer networks, 4th edn. Prentice Hall, Upper Saddle River (2003)
37. Gangwal, O., Rădulescu, A., Goossens, K., González Pestana, S., Rijpkema, E.: Building predictable systems on chip: An analysis of guaranteed communication in the Æthereal network on chip. Dynamic and Robust Streaming in and between Connected Consumer-Electronic Devices, 1–36 (2005)
38. Radulescu, A., Dielissen, J., Pestana, S.G., Gangwal, O.P., Rijpkema, E., Wielage, P., Goossens, K.: An efficient on-chip NI offering guaranteed services, shared-memory abstraction, and flexible network configuration. Comput.-Aided Des. Integr. Circuits Syst. IEEE Trans. **24**(1), 4–17 (2005)
39. Millberg, M., Nilsson, E., Thid, R., Kumar, S., Jantsch, A.: The Nostrum backbone-a communication protocol stack for networks on chip. In: Proceedings of the VLSI Design on 17th International Conference, IEEE, pp. 693–696 (2004)
40. Radulescu, A., Goossens, K.: Communication services for networks on chip. Domain-Specific Processors: Systems, Architectures, Modeling, and Simulation, 193–213 (2004)
41. Radulescu, A., Dielissen, J., Pestana, S.G., Gangwal, O.P., Rijpkema, E., Wielage, P., Goossens, K.: An efficient on-chip NI offering guaranteed services, shared-memory abstraction, and flexible network configuration. IEEE Trans. Comput. Aided Des. Integr. Circuits Syst. **24**(1), 4–17 (2005)
42. Goossens, K., Dielissen, J., Radulescu, A.: Æthereal network on chip: concepts, architectures, and implementations. Des. Test Comput. IEEE **22**(5), 414–421 (2005)
43. Scherrer, A., Fraboulet, A., Risset, T.: Hardware wrapper classification and requirements for on-chip interconnects. In: Signaux, Circuits et Systèmes 2004, p. 4
44. Chapiro, D.M.: Globally-Asynchronous Locally-Synchronous Systems. In. Stanford Univ CA Dept of Computer Science (1984)
45. Dally, W.J., Seitz, C.L.: Deadlock-free message routing in multiprocessor interconnection networks. (1988)
46. Dally, W.J., Towles, B.: Route packets, not wires: On-chip interconnection networks. In: Proceedings of the Design Automation Conference, IEEE, pp. 684–689 (2001)
47. Feige, U., Raghavan, P.: Exact analysis of hot-potato routing. In: Proceedings of the Foundations of Computer Science, 33rd Annual Symposium, IEEE, pp. 553–562 (1992)
48. Fleury, E., Fraigniaud, P.: A general theory for deadlock avoidance in wormhole-routed networks. IEEE Trans. Parallel Distrib. Syst. **9**(7), 626–638 (1998)
49. Ciordas, C., Basten, T., Radulescu, A., Goossens, K., Meerbergen, J.: An event-based network-on-chip monitoring service. In: Proceedings of the High-Level Design Validation and Test Workshop on Ninth IEEE International, pp. 149–154 (2004)
50. Sepulveda, J., Flórez, D., Immler, V., Gogniat, G., Sigl, G.: Efficient security zones implementation through hierarchical group key management at NoC-based MPSoCs. Microprocess. Microsyst. **50**, 164–174 (2017)
51. Fiorin, L., Silvano, C., Sami, M.: Security aspects in networks-on-chips: Overview and proposals for secure implementations. In: Proceedings of the Digital System Design Architectures, Methods and Tools. DSD 2007 on 10th Euromicro Conference, IEEE, pp. 539–542 (2007)
52. Fiorin, L., Palermo, G., Lukovic, S., Catalano, V., Silvano, C.: Secure memory accesses on networks-on-chip. Comput. IEEE Trans. **57**(9), 1216–1229 (2008)
53. Kapoor, H.K., Rao, G.B., Arshi, S., Trivedi, G.: A security framework for noc using authenticated encryption and session keys. Circuits Syst. Sign. Process. **32**(6), 2605–2622 (2013)
54. Baron, S., Wangham, M.S., Zeferino, C.A.: Security mechanisms to improve the availability of a Network-on-Chip. In: Proceedings of the Electronics, Circuits, and Systems (ICECS) on IEEE 20th International Conference, pp. 609–612 (2013)
55. Ghofrani, A., Parikh, R., Shamshiri, S., DeOrio, A., Cheng, K.-T., Bertacco, V.: Comprehensive online defect diagnosis in on-chip networks. In: VTS, pp. 44–49 (2012)
56. Babaei, S., Mansouri, M., Aghaei, B., Khadem-Zadeh, A.: Online-structural testing of routers in network on chip. World Applied Sci. J. **14**(9), 1374–1383 (2011)
57. Alaghi, A., Karimi, N., Sedghi, M., Navabi, Z.: Online NoC switch fault detection and diagnosis using a high level fault model. In: Proceedings of the 22nd IEEE International Symposium on Defect and Fault-Tolerance in VLSI Systems (DFT 2007), IEEE, pp. 21–29 (2007)
58. Alamian, S.S., Fallahzadeh, R., Hessabi, S., Alirezaie, J.: A novel test strategy and fault-tolerant routing algorithm for NoC routers. In: Proceedings of the 17th CSI International Symposium on Computer Architecture & Digital Systems (CADS 2013), IEEE, pp. 133–136 (2013)
59. Cota, É., de Morais Amory, A., Lubaszewski, M.S.: Test and diagnosis of routers. In: Cota, É. (ed.) Reliability. Availability and serviceability of networks-on-chip, pp. 115–132. Springer, New York (2012)
60. Hosseinabady, M., Dalirsani, A., Navabi, Z.: Using the inter-and intra-switch regularity in NoC switch testing. In: Proceedings of the conference on Design, automation and test in Europe, pp. 361–366. EDA Consortium (2007)
61. Nazari, M., Zolfy Lighvan, M., Daie Koozekonani, Z., Sadeghi, A.: a novel HW/SW based NoC router self-testing methodology. arXiv:1609.04569 (2016)
62. Nazarian, G.: On-line testing of routers in networks-on-chip. Delft University of Technology, Delft (2008)
63. Aghaei, B., Khademzadeh, A., Reshadi, M., Badie, K.: Link testing: a survey of current trends in network on chip. J. Electron. Test. **33**, 209–225 (2017)
64. Aghaei, B., Badie, K., Khademzadeh, A., Reshadi, M.: The cost-effective fault detection and fault location approach for communication channels in NoC. J. Supercomput. **73**, 5034–5052 (2017)

65. Aghaei, B., Khademzadeh, A., Reshadi, M., Badie, K.: A new BIST-based test approach with the fault location capability for communication channels in network-on-chip. J. Electron. Test. **33**, 501–513 (2017)
66. Aghaei, B.: A high fault coverage test approach for communication channels in network on chip. Microelectron. Reliab. **75**, 178–186 (2017)
67. Aghaei, B., Babaei, S.: The new test wrapper design for core testing in Packet-Switched Micro-Network on Chip. In: Proceedings of the 2nd International Conference on Power Electronics and Intelligent Transportation System (PEITS), pp. 19–20 (2009)
68. Amory, A.M., Goossens, K., Marinissen, E.J., Lubaszewski, M., Moraes, F.: Wrapper design for the reuse of a bus, network-on-chip, or other functional interconnect as test access mechanism. Comput Digit Tech IET **1**(3), 197–206 (2007)
69. Cota, É., Carro, L., Lubaszewski, M.: Reusing an on-chip network for the test of core-based systems. ACM Trans Des Autom Electron Syst (TODAES) **9**(4), 471–499 (2004)
70. Xiang, D., Zhang, Y.: Cost-effective power-aware core testing in NoCs based on a new unicast-based multicast scheme. IEEE Trans. Comput. Aided Des. Integr. Circuits Syst. **30**(1), 135–147 (2011)
71. Fiorin, L., Sami, M.: Fault-Tolerant Network Interfaces for Networks-on-Chip. Depend Secure Comput. IEEE Trans. **11**(1), 16–29 (2014)
72. Thonnart, Y., Beigné, E., Vivet, P.: Design and implementation of a GALS adapter for ANoC based architectures. In: Proceedings of the Asynchronous Circuits and Systems on ASYNC'09 15th IEEE Symposium, IEEE, pp. 13–22 (2009)
73. Matos, D., Carro, L., Susin, A.: Associating packets of heterogeneous cores using a synchronizer wrapper for NoCs. In: Proceedings of the Circuits and Systems (ISCAS) on IEEE International Symposium, IEEE, pp. 4177–4180 (2010)
74. Fattah, M., Daneshtalab, M., Liljeberg, P., Plosila, J.: Transport layer aware design of network interface in many-core systems. In: Proceedings of the Reconfigurable Communication-centric Systems-on-Chip (ReCoSoC) on 7th International Workshop, IEEE, pp. 1–7 (2012)
75. Lee, S.E., Bahn, J.H., Yang, Y.S., Bagherzadeh, N.: A generic network interface architecture for a networked processor array (NePA). In: Proceedings of the International Conference on Architecture of Computing Systems, pp. 247–260. Springer (2008)
76. Sparsø, J., Kasapaki, E., Schoeberl, M.: An area-efficient network interface for a TDM-based network-on-chip. In: Proceedings of the Conference on Design, Automation and Test in Europe, pp. 1044–1047. EDA Consortium (2013)
77. Ruaro, M., Lazzarotto, F.B., Marcon, C.A., Moraes, F.G.: DMNI: A specialized network interface for NoC-based MPSoCs. In: Proceedings of the Circuits and Systems (ISCAS) on IEEE International Symposium, IEEE, pp. 1202–1205 (2016)
78. Bjerregaard, T., Mahadevan, S., Olsen, R.G., Sparso, J.: An OCP compliant network adapter for GALS-based SoC design using the MANGO network-on-chip. In: Proceedings of the System-on-Chip on International Symposium, IEEE, pp. 171–174 (2005)
79. Bjerregaard, T., Sparso, J.: A router architecture for connection-oriented service guarantees in the MANGO clockless network-on-chip. In: Proceedings of the Design, Automation and Test in Europe, IEEE, pp. 1226–1231 (2005)
80. Fiorin, L., Palermo, G., Silvano, C.: A security monitoring service for NoCs. In: Proceedings of the 6th IEEE/ACM/IFIP International Conference on Hardware/Software codesign and system synthesis, ACM, pp. 197–202 (2008)
81. Xia, B., Wu, K., Xiang, C., Yang, M., Liu, P., Yao, Q.: Network interface design based on mutual interface definition. Int. J. High Perform. Syst. Archit. **2**(3–4), 168–176 (2010)
82. Attia, B., Wissem, C., Noureddine, A., Zitouni, A., Torki, K., Tourki, R.: A new pipelined network interface for Network on Chip with latency and jitter optimization. In: Proceedings of the Microelectronics (ICM) on International Conference, IEEE, pp. 1–6 (2011)
83. Chouchene, W., Attia, B., Zitouni, A., Abid, N., Tourki, R.: A low power network interface for network on chip. In: Proceedings of the Systems, Signals and Devices (SSD) on 8th International Multi-Conference, IEEE, pp. 1–6 (2011)
84. Swaminathan, K., Lakshminarayanan, G., Ko, S.-B.: Design and verification of an efficient WISHBONE-based network interface for network on chip. Comput. Electr. Eng. **40**(6), 1838–1857 (2014)
85. Bhojwani, P., Mahapatra, R.: Interfacing cores with on-chip packet-switched networks. In: Proceedings of the VLSI Design on 16th International Conference, IEEE, pp. 382–387 (2003)
86. Lai, Y.-L., Yang, S.-W., Sheu, M.-H., Hwang, Y.-T., Tang, H.-Y., Huang, P.-Z.: A high-speed network interface design for packet-based NoC. In: Proceedings of the Communications, Circuits and Systems on International Conference, IEEE, pp. 2667–2671 (2006)
87. Yang, X., Qing-li, Z., Fang-fa, F., Ming-yan, Y., Cheng, L.: NISAR: An AXI compliant on-chip NI architecture offering transaction reordering processing. In: Proceedings of the ASIC. ASICON'07. 7th International Conference, IEEE, pp. 890–893 (2007)
88. Ebrahimi, M., Daneshtalab, M., Liljeberg, P., Plosila, J., Tenhunen, H.: A high-performance network interface architecture for NoCs using reorder buffer sharing. In: Proceedings of the Parallel, Distributed and Network-Based Processing (PDP), 2010 18th Euromicro International Conference, IEEE, pp. 546–550 (2010)
89. Daneshtalab, M., Ebrahimi, M., Plosila, J., Tenhunen, H.: CARS: Congestion-aware request scheduler for network interfaces in NoC-based manycore systems. In: Proceedings of the Conference on Design, Automation and Test in Europe, pp. 1048–1051. EDA Consortium (2013)
90. Tran, X.-T., Nguyen, T., Phan, H.-P., Bui, D.-H.: AXI-NoC: High-Performance Adaptation Unit for ARM Processors in Network-on-Chip Architectures. IEICE Trans. Fund. Electron. Commun. Comput. Sci. **100**(8), 1650–1660 (2017)
91. Hu, J., Marculescu, R.: Energy-aware mapping for tile-based NoC architectures under performance constraints. In: Proceedings of the 2003 Asia and South Pacific Design Automation Conference, ACM, pp. 233–239 (2003)
92. Vangal, S.R., Howard, J., Ruhl, G., Dighe, S., Wilson, H., Tschanz, J., Finan, D., Singh, A., Jacob, T., Jain, S.: An 80-tile sub-100-w teraflops processor in 65-nm cmos. IEEE J. Solid-State Circuits **43**(1), 29–41 (2008)
93. Bhojwani, P., Mahapatra, R.N.: Core network interface architecture and latency constrained on-chip communication. In: Proceeding of the Quality Electronic Design on ISQED'06. 7th International Symposium, IEEE, pp. 6–363 (2006)
94. Ost, L., Mello, A., Palma, J., Moraes, F., Calazans, N.: MAIA: a framework for networks on chip generation and verification. In: Proceedings of the 2005 Asia and South Pacific Design Automation Conference, ACM, pp. 49–52 (2005)
95. Fiorin, L., Palermo, G., Lukovic, S., Silvano, C.: A data protection unit for NoC-based architectures. In: Proceedings of the 5th IEEE/ACM international conference on Hardware/software codesign and system synthesis, ACM, pp. 167–172 (2007)

96. Pande, P.P., Grecu, C., Ivanov, A., Saleh, R., De Micheli, G.: Design, synthesis, and test of networks on chips. IEEE Des. Test Comput. **22**(5), 404–413 (2005)
97. Furber, S., Bainbridge, J.: Future trends in SoC interconnect. In: Proceedings of the System-on-Chip, International Symposium, IEEE, pp. 183–186 (2005)
98. Tatas, K., Siozios, K., Soudris, D., Jantsch, A.: NoC Verification and Testing. In: Jantsh, A. (ed.) Designing 2D and 3D network-on-chip architectures, pp. 147–159. Springer, New York (2014)
99. Grammatikakis, M.D., Papadimitriou, K., Petrakis, P., Papagrigoriou, A., Kornaros, G., Christoforakis, I., Tomoutzoglou, O., Tsamis, G., Coppola, M.: Security in MPSoCs: a NoC firewall and an evaluation framework. IEEE Trans. Comput. Aided Des. Integr. Circuits Syst. **34**(8), 1344–1357 (2015)
100. Compiler, S.D.: Synopsys Corporation. In. (1999)

Babak Aghaei He received the B.Sc. degree in Computer Engineering-Hardware from Shomal University, Amol, Iran in 2007 and the M.Sc. degree in Computer Engineering-Computer Systems Architectures from Islamic Azad University, Tabriz Branch, Iran in 2009. He also received his Ph.D. degree in Computer Engineering-Hardware from Science and Research Branch of Islamic Azad University (SRBIAU), Tehran, Iran in 2017. He is currently an Assistant Professor in Department of Computer Engineering, Malekan Branch, Islamic Azad University, Malekan, Iran. His research interests include, Design for Test (DfT), fault tolerant, reliability, and routing of embedded systems and on-chip interconnection networks (NoC).

Midia Reshadi Received his M.Sc. degree in computer architecture from Science and Research Branch of Islamic Azad University (SRBIAU), Tehran, Iran in 2005. He also received his Ph.D. degree in computer architecture from SRBIAU, Tehran, Iran in 2010. He is currently Assistant Professor in Faculty of Electrical and Computer Engineering of SRBIAU. His research interests include Photonic NoCs, fault and yield issues in NoCs, routing and switching in on-chip communication networks. He is a member of IEEE.

Mohammad Masdari Hi is an Assistant Professor in the Department of Computer Engineering, Urmia Branch, Islamic Azad University, Urmia Iran. He received his Ph.D. degree in 2010, from the School of Computer Engineering at science and research University Tehran branch.

Seyyed Hadi Sajadi He received his B.Sc. from the Department of Electrical and Computer Engineering of Shahid Beheshti University and his MSc on Economical-Social System Engineering from Mazandaran University of Science and Technology. He is currently a Ph.D. candidate in Software Engineering at Sharif University of Technology International Campus. His research interests include ethics in cyber-space, child online protection and also leveraging ICT in social applications, especially the evolution of social norms using online social.

Mehdi Hosseinzadeh He was born in Dezful, a city in the southwestern Iran; in 1981 he received his B.Sc. in Computer Hardware Engineering from the Islamic Azad University, Dezful branch, Iran in 2003. He also received M.Sc. and Ph.D. degrees in Computer Systems Architecture from the Science and Research Branch, Islamic Azad University, Tehran, Iran in 2005 and 2008, respectively. He is currently an Assistant Professor in the University of Medical Science, Tehran, Iran. His research interests are Computer Arithmetics, with emphasis on Residue Number System, Cryptography, Network Security, Radio Frequency Identification (RFID) systems and E-Commerce. He has authored and co-authored many high-cited scientific articles published in peer-reviewed conference proceedings and international journals.

Aso Darwesh He received the B.Sc. in Mathematics in University of Sulaimani, Iraq 2001, M.S. degrees in Computer Science in University of Rene Descartes, France 2007 and Ph.D. in Computer Science, University of Pierre and Mari Curie, France 2010. Currently, he is Associate Professor in the Information Technology Department, University of Human Development, Sulaimani, Iraq. His research interests include Serious Games, Adaptive Learning Cognitive Diagnosis in E-Learning, Learning Systems, Computer Networks, Networking Security, and Data Mining.